Escuela Politécnica Superior

Aplicación para facilitar la localización y ocupación de plazas de aparcamiento para discapacitados

Máster Universitario en Desarrollo de Software para Dispositivos Móviles

Trabajo Fin de Máster

Autor:
Sergio Muñoz Gómez

Tutor:
Miguel Ángel Lozano Ortega

Septiembre 2020

Universitat d'Alacant
Universidad de Alicante

Resumen

El trabajo consta en el desarrollo de un sistema que facilite a las personas con problemas de movilidad el duro trabajo que para ello supone el simple hecho de desplazarse con el coche, debido a que en muchos sitios no está bien indicado o no disponen de plazas de aparcamiento para minusválidos. Esto supone que estas personas necesiten un tiempo previo de planificación y organización es sus viajes para ver si tienen facilidad de aparcamiento en el lugar allá donde se dirijan.

Para solucionar estos problemas cotidianos, el grupo de Artefactos de la Universidad de Alicante, la Asociación para la integración socio laboral de personas con discapacidad física y sensorial (AMFI), la aportación de otro TFM del que parte este y yo, hemos continuado con el proyecto de EnableParking. Este sistema permite a los usuarios buscar por un mapa o buscando una dirección aquellos lugares donde haya plazas para minusválidos, así como su disponibilidad. A parte de lo anterior, si el usuario decide registrarse podrá disponer de mayores funcionalidades como la gestión de su perfil, el registro manual y automático de su aparcamiento. Este sistema de registro de aparcamiento permite actualizar con mayor precisión el estado de las plazas, así como permitir al usuario encontrar la plaza donde haya aparcado con mayor facilidad.

Agradecimientos

A mi familia, por permitirme vivir en otra ciudad y enseñarme a valerme por mí mismo.

A mis amigos y compañeros, por ayudarme y apoyarme en todos aquellos momentos en los que necesitaba una mano que coger.

A mi tutor, por todo lo que me ha enseñado y guiado en todo el proceso de desarrollo de este proyecto, incluso en momento en los que van más allá de su horario laboral.

Gracias.

Índice de contenido

Resumen *1*

Agradecimientos *2*

1. Introducción y justificación del proyecto *3*

2. Estado de la cuestión *4*

2.1. Estado actual de la aplicación **4**

2.2. Funcionamiento de las tarjetas de movilidad reducida **4**

2.3. Tecnologías aplicables a la detección automática de ocupación de plazas **6**

3. Metodología *12*

4. Objetivos *14*

5. Análisis y especificación *15*

5.1. Actores **15**

5.2. Requisitos de usuario **15**

5.2.1. Requisitos de usuario del actor usuario no registrado 15

5.2.2. Requisitos de usuario del actor usuario registrado 16

5.3. Requisitos funcionales **17**

5.4. Requisitos no funcionales **24**

6. Diseño *29*

6.1. Arquitectura lógica **29**

6.2. Arquitectura física **30**

6.3. Modelo Entidad - Relación (E-R) **32**

6.4. Diseño de la interfaz **33**

6.4.1. Pantalla principal 34

6.4.2. Pantalla detalles de aparcamiento 36

6.4.3. Pantalla de registro de usuario 37

6.4.4. Pantalla login 38

6.4.5. Pantalla perfil 39

6.5. Diagramas de secuencia **40**

6.6. Guía de estilos **41**

7. Implementación *42*

7.1. Servidor **42**

7.1.1. Módulo Cloud Firestore 43

7.1.2. Módulo Cloud Functions 44

7.1.3. Módulo de Cloud Storage 45

7.1.4. Módulo de Google Analytics 46

7.1.5. Módulo de Cloud Messaging 46

7.2. Comunicación con las balizas 47
7.2.1. Requisitos de implementación 48
7.2.2. Formato del protocolo AltBeacon 49
7.2.3. Diagrama del protocolo 49
7.2.4. Campos del protocolo 50
7.2.5. Ejemplo de implementación de AltBeacons 52
7.3. Componentes implementados 53
7.3.1. Actividades 53
7.3.2. Clases del modelo de datos 61
7.3.3. Receiver 61
7.3.4. Servicio de notificaciones 62
7.3.5. Funciones en la nube 62
7.4. Tecnologías utilizadas 63
7.5. Herramientas utilizadas 66
8. Conclusiones y líneas futuras 68
8.1. Conclusiones 68
8.2. Líneas futuras 69
9. Webgrafía 71

Índice de figuras

Figura 1 - Logo Ingepark 7
Figura 2 - Sistema Ingepark 7
Figura 3 - Sistema EsquinsaParking 8
Figura 4 - Sistema 3PGS OUTDOOR 9
Figura 5 - Sistema PMUS 11
Figura 6 - Metodología ágil SCRUM 12
Figura 7 - Tablero tareas SCRUM Github 13
Figura 8 - Diagrama de casos de uso del actor Usuario no registrado 16
Figura 9 - Diagrama de casos de uso del actor Usuario registrado 17
Figura 10 - Arquitectura lógica 30
Figura 11 - Arquitectura física 31
Figura 12 - Diagrama Entidad/Relación 32
Figura 13 - Boceto pantalla principal 34
Figura 14 - Boceto pantalla detalles de aparcamiento 36
Figura 15 - Boceto pantalla de registro de usuario 37
Figura 16 - Boceto pantalla login 38
Figura 17 - Boceto pantalla perfil 39
Figura 18 - Diagrama de secuencia del registro automático de la ocupación 40
Figura 19 - Guía de colores 41
Figura 20 - Estructura del AltBeacon Advertisement 49
Figura 21 - Diagrama del protocolo AltBeacon 49
Figura 22 - Estructura de Android Beacon Library 65

1. Introducción y justificación del proyecto

La elección de elegir este proyecto surgió de la necesidad de realizar un TFM cuyo resultado fuera una aplicación que fuera útil y llamativa para la gente.

Dentro de las varias propuestas que había he escogido este proyecto porque el tema referente a la ayuda a los discapacitados me llama especialmente la atención, debido a que siempre es más útil una aplicación cuando existe una verdadera necesidad.

Además de lo mencionado, también fue un factor de influencia el hecho de que realizar este proyecto me permitiría trabajar con una entidad externa, concretamente con el equipo del Grupo Artefactos de la UA. Esto es debido a que el proyecto no parte de cero, sino que continúo un proyecto que empezó un compañero del máster el año pasado. El proyecto del compañero consistía en una aplicación Android que permita a los usuarios saber dónde pueden disponer de una plaza de aparcamiento para personas con movilidad reducida allá donde quieran buscar.

Partiendo de este proyecto, la idea es crear una aplicación Android que implemente las funcionalidades ya existentes de localización de plazas de aparcamientos para minusválidos junto con otras que faciliten aún más saber la disponibilidad de estas y la concienciación de los conductores de no ocupar las plazas de forma indebida. Para ello vamos a utilizar balizas situadas en las plazas de aparcamiento para minusválidos que detecten si está libre o no.

2. Estado de la cuestión

En este capítulo revisaremos en primer lugar el estado de dicha aplicación. Tras ello, explicaremos el funcionamiento de las tarjetas de movilidad reducida que se utiliza actualmente para controlar el aparcamiento en dichas plazas, y finalizaremos analizando diferentes tecnologías aplicables a la detección automática de ocupación de plazas.

2.1. Estado actual de la aplicación

Como ya se ha comentado en la introducción, este proyecto no parte de cero, sino que ya contamos con una aplicación Android para la localización de plazas de aparcamiento de minusválidos. Para ello disponemos una pantalla principal en la que se nos muestra un mapa donde aparecen las localizaciones de todas las plazas de aparcamiento registradas permitiéndonos ver su dirección y el estado de su ocupación. Si pulsamos en la etiqueta, vemos otra pantalla con los datos del aparcamiento y la posibilidad de votarla, navegar hacia ella, y poder avisar sobre que la plaza ya no está disponible. A parte de esto, en la pantalla principal, también podemos sugerir nuevas plazas de aparcamiento. Tanto las sugerencias de nueva plaza como los avisos de plaza no existente son controlados por un administrador que tiene acceso al servidor, el cual crea o destruye plazas en la base de datos después de verificar que la información recibida es veraz.

Para conseguir aumentar las funcionalidades voy a trabajar con el equipo de artefactos de la Universidad de Alicante para poder hacer uso de las balizas y trabajar con ellas.
Antes de la decisión del uso de balizas para determinar la disponibilidad de la plaza de aparcamiento, se estuvieron estudiando otras opciones como el uso de la ubicación del propio móvil de los usuarios, pero estas fueron desechadas por la poca precisión y versatilidad que ofrecían. También se habló sobre el uso de otros sensores que mejorase la comunicación, pero se descartaron, de momento, por el coste que supondría su obtención, como por ejemplo sensores magnéticos o cámaras.

Para trabajar con las balizas se va a realizar una conexión bluetooth entre el móvil y la propia baliza a través de la emisión de Beacons.

2.2. Funcionamiento de las tarjetas de movilidad reducida

Todo este proyecto se basa en la gestión de plazas de aparcamiento para minusválidos y en las tarjetas de aparcamiento para personas de movilidad reducida, o tarjetas de movilidad reducida para abreviar, que se otorgan a las personas con esta minusvalía. Por este motivo es muy importante realizar un trabajo de investigación sobre el tema para entender cómo funciona toda esta gestión.

Respecto al primer aspecto de la gestión de las plazas de aparcamiento de minusválidos disponemos de la memoria pública del compañero que comenzó este proyecto. Resumiendo, su referencia se puede decir que hoy en día la búsqueda de este tipo de plazas de aparcamiento es una tarea ardua ya que los organismos públicos encargados de gestionar dicha información, los ayuntamientos de los municipios, en general, no dan demasiada información sobre el tema ya que no suelen disponer de un listado de plazas con información sobre las mismas.

Respecto a la parte de la gestión de las tarjetas de movilidad reducida tenemos que empezar diciendo que la solicitud de estas depende del ayuntamiento al que se pertenezca, por ese motivo, actualmente se tiene cierta libertad en los requerimientos para solicitar y conceder una, aunque en líneas generales se exigen dos requisitos para conseguirla:

- Acreditar un grado de discapacidad igual o superior al 33%. (por lo tanto, necesitarás tu certificado de discapacidad)
- Tener un baremo de movilidad reducida positivo

La tarjeta de aparcamiento para discapacitados se adjudica a personas con discapacidad física. El resto de las diversidades funcionales (intelectuales, auditivas etc.) no pueden acceder a ella. Su uso es personal e intransferible, aunque puede utilizarse también cuando el titular es usuario del coche, aunque no sea su conductor.

Debido a que la gestión no está centralizada, la información que se muestra en ellas no está estandarizada, aunque hay datos que sí están presente en todas ellas como son la fecha de caducidad, nº de tarjeta y el logo indicando que es admisible en toda la Unión Europea.

Las personas con movilidad reducida en posesión de una tarjeta de aparcamiento para discapacitados tienen derecho a solicitar y obtener una plaza de aparcamiento reservada de forma personal. Las condiciones particulares para esta reserva dependen de cada ayuntamiento, que decidirá cuántas de ellas se destinan a este fin y quién tiene prioridad para conseguirlas.

También serán ellos los encargados de establecer, en las condiciones que así consideren, la creación de plazas de aparcamiento para discapacitados a petición concreta de un solicitante.

Se observan dos tipos de plazas de aparcamiento para discapacitados que pueden reservarse:

- De origen: aquellas que se sitúan cerca del domicilio de la persona discapacitada.
- De destino: Ubicándose normalmente en recintos de utilización común tales como edificios públicos, centros sanitarios, docentes, deportivos, comerciales, de ocio, etc.

En ambos casos el titular de la tarjeta deberá ser el conductor del vehículo y presentar la documentación respecto a la vigencia y validez de esta. La concesión de las segundas se centrará en la necesidad de la misma para acceder al área de trabajo. En este caso se deberá añadir a la documentación anterior un certificado de empresa acreditando el lugar de trabajo.

2.3. Tecnologías aplicables a la detección automática de ocupación de plazas

Como ya he comentado en la introducción, partimos del proyecto de un compañero al cual quiero añadirle más funcionalidades. La funcionalidad más importante añadida, y en la que se basa esta contribución al proyecto, es la gestión de la ocupación de la plaza y la relación de ésta con los usuarios de la aplicación. Por este motivo es importante analizar el mercado para ver qué otros sistemas de gestión de plazas existen, en qué consisten y cómo funcionan:

- **Ingepark:** El sistema de guiado de parking INGEPARK (11) es una solución ideal para una eficaz gestión de las plazas de aparcamiento. A través de señales, tales como flechas de guiado e indicadores luminosos, el usuario verá reducido sensiblemente su tiempo de búsqueda de aparcamiento. Así mismo, en el puesto de control, el gestor del parking dispondrá de un software en el que visualizará la situación real de las plazas, pudiendo controlar tiempos de ocupación, reservar plazas para eventos, visualización de estadísticas de ocupación por zonas, etc.

    - Sistema de guiado: La detección supone la parte básica del sistema de guiado, ya que la gestión comienza con la lectura que se realiza sobre el estado de cada plaza (Libre-Ocupado).
      En el momento en que el sensor detecta que una plaza cambia de estado (Libre/Ocupado), la señaliza y además, envía la información al software de gestión, de manera que el gestor siempre tiene conocimiento real del estado de las plazas. Cada grupo de plazas en su acceso tiene asignada una flecha indicadora, de forma que en cuanto se ocupan todas sus plazas la flecha se apaga, indicando al usuario que no dispone de plazas libres en esa zona. De igual manera, en los accesos a las plantas o zonas principales se colocan indicadores numéricos con las plazas disponibles para dar al usuario una idea de la ocupación.

    - Software de gestión: INGEPARK está gestionado por un software basado en entorno Windows para su fácil manejo. El software controla la lectura de cada sensor y regula su funcionamiento.
      Si se desea, se pueden dividir las plazas en zonas, y asignar visualmente colores distintivos a cada una. De esta manera se consigue un control visual del estatus del parking. También se puede consultar la información de ocupación numéricamente, visualizando el número de plazas ocupadas y libres en cada una de las zonas y/o plantas de las que disponga el parking.

Figura 1 - Logo Ingepark

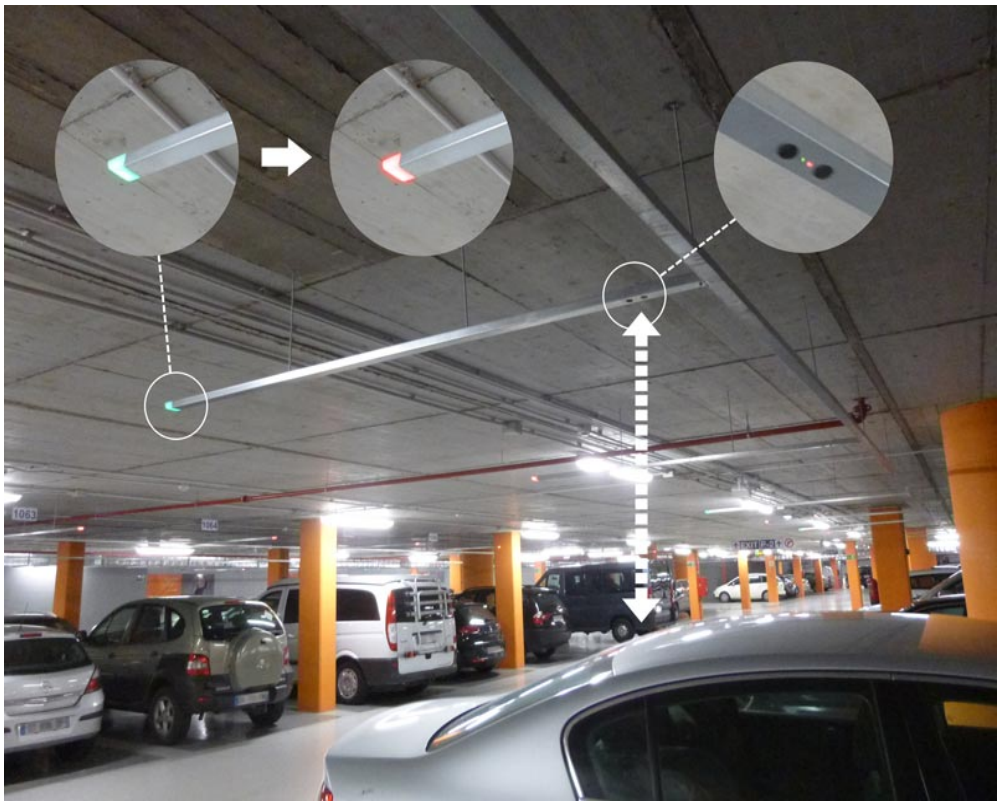


Figura 2 - Sistema Ingepark

- **EquinsaParking:** Dispone del sistema GUIA (Gestión de Usuarios en Interior de Aparcamientos) (12) que permite guiar al cliente del aparcamiento mediante señalizaciones de ocupación por plaza y carteles indicadores. Asimismo, pone a disposición del explotador del aparcamiento información de gran utilidad sobre el estado de ocupación de las plazas.

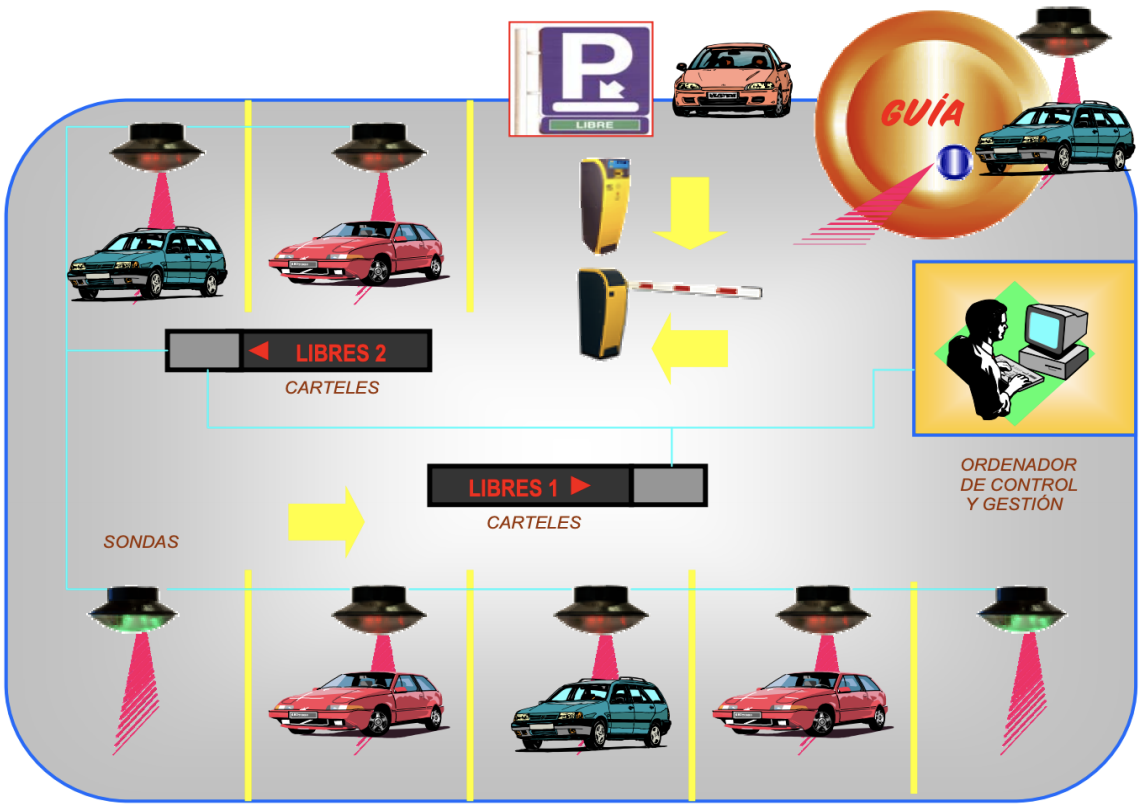


Figura 3 - Sistema EsquinsaParking

La detección de ocupación es realizada por sondas situadas en cada plaza de aparcamiento basadas en ondas de ultrasonidos emitidas y reflejadas.
Las sondas se encuentran ubicadas en cada plaza en el techo del aparcamiento e incorporan señalización óptica de dos colores que informan al usuario del estado de la plaza. El color verde se utiliza para indicar que la plaza está libre y el color rojo para el estado de ocupada. Debido a la utilización de leds de alta luminosidad, la señalización puede observarse a larga distancia. No obstante, pueden además utilizarse extensiones de señalización ubicadas en la cabecera de cada plaza, permitiendo una mayor visibilidad de la señalización de plaza.
Este concepto de guía en aparcamiento permite:

- Mejorar la calidad del servicio ofrecido al usuario.
- Aparcar en menos tiempo.
- Disminuir por tanto el nivel de contaminación.
- Incrementar la satisfacción del cliente.

Los Carteles indicadores informan al usuario de la disponibilidad de plazas libres, guiándolos además hasta ellas mediante flechas.
La información suministrada por los paneles sobre las sondas o las extensiones y el estado de funcionamiento de los Carteles indicadores, es almacenada en la base de datos del ordenador de Control y Gestión, consintiendo así disponer de un histórico de todos los movimientos de ocupación de las plazas del aparcamiento.

El sistema GUIA permite al explotador del aparcamiento disponer de una valiosa herramienta de estadísticas sobre el nivel de ocupación, la generación automática de listados de plazas con tiempos de aparcamiento superior al umbral de horas determinado por el gestor, el listado automático de vehículos estacionados a partir de una determinada hora en la noche (facilitando la toma de datos al servicio de seguridad del aparcamiento).
Por otro lado, también pueden ser bloqueadas desde el ordenador de Control y Gestión las plazas a ocupado o libre, permitiendo aparcar los vehículos por zonas, con el consiguiente ahorro energético de las zonas no ocupadas cuando la demanda de plazas así lo permita. El sistema GUIA constituye así un Sistema de Guiado de Usuarios (vehículos) en el Interior de aparcamientos, es modular y puede configurarse de acuerdo con las necesidades del cliente.

Tanto esta herramienta como la anterior están pensadas para la gestión de parkings, usualmente, privados como pueden tener los centros comerciales, supermercados, etc.

- **3PGS OUTDOOR:** Es un sistema de gestión de aparcamiento en vía pública que permite un control automático de plazas y un guiado de plazas en la zona azul (13).
  Este sistema de control automático de plazas funciona mediante tecnología wireless, todos los elementos que lo forman están conectados a la misma red. Cada detector de plazas zona azul informa del estado de su plaza a la controladora de zona, que a su vez transmite la información al parquímetro, bajo el software de gestión de plazas. A esa red están conectadas las señales de aparcamiento áreas urbanas y los propios controladores que realizan el control de plazas en la calle y la gestión de plazas en vía pública.
  Gracias a este sistema de guiado de plazas y gestión aparcamiento público automático, los usuarios podrán conocer las plazas libres disponibles en la vía pública y conducir directamente hacia ellas, reduciendo el tráfico y ahorrando tiempo.

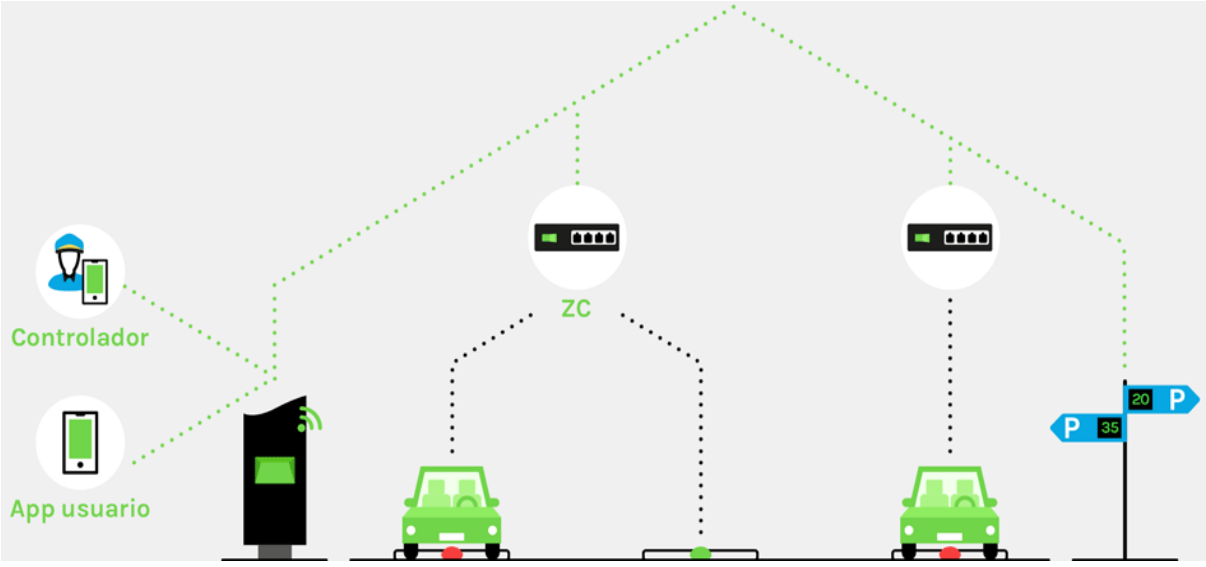


Figura 4 - Sistema 3PGS OUTDOOR

Características clave:

- **Sensor magnético:** Un pequeño detector colocado en cada plaza que detecta si hay un vehículo aparcado o está libre, indicándose con una luz LED.
- **Señal LED:** De forma opcional, se pueden instalar diferentes señales LED que indicar el número de plazas libres en una calle.
- **Controladora de la zona:** Controladora a la que se conectan todos los detectores magnéticos, controlando el número de plazas de un área concreta.
- **Parquímetro:** Encargado de procesar todas las plazas de aparcamiento en la vía pública. Recibe los datos de las diferentes controladoras de zona.
- **App controlador:** El controlador de la vía pública cuenta con una app desde la que gestionar todas las infracciones.
- **App usuario:** De forma opcional, los usuarios podrán utilizar una app en sustitución del parquímetro y ver las plazas disponibles.

- **PMUS, Sistema de gestión y control de aparcamiento en vía pública:** El proyecto consiste en el suministro, instalación y puesta en marcha de los sistemas y tecnologías necesarios para la gestión y operación de estacionamiento en la vía pública (14), con capacidad de adaptación tanto a las necesidades actuales como futuras, capaz de gestionar distinto tipos de plazas de estacionamiento catalogadas en función de su uso: Carga y descarga, PMRs, detección plazas no autorizadas, guiado a plazas libres y control de plazas de aparcamiento entre otras. Esta actuación se centra en la gestión y control del estacionamiento de vehículos en plazas reservadas a personas con movilidad reducida en diferentes zonas de Valencia. La gestión y control del estacionamiento de vehículos en plazas reservadas para carga y descarga en la calle Azorín y en la zona de aparcamiento ubicada en la calle Reverendo Padre José Palacios, el tramo comprendido entre la calle Roll de les Eres y la calle Marques del Turia. La gestión y control del estacionamiento de vehículos en la zona de aparcamiento ubicada junto a la biblioteca en la calle Marqués del Turia. En esta actuación quedarán gestionadas y controladas 7 plazas reservadas a personas con movilidad reducida, 2 zonas de carga y descarga disponiendo en cada zona dos plazas de estacionamiento de vehículos y 94 plazas de estacionamiento de vehículos.

Figura 5 - Sistema PMUS

Tanto este proyecto como el anterior, y a diferencia de los 2 primeros, están orientados al área urbana, es decir, como nuestro proyecto, serían gestionados por el ayuntamiento que lo establezca.

3. Metodología

Una buena planificación es siempre un aspecto clave en el ciclo de vida de un proyecto, sobretodo cuando el tiempo es limitado. Debido a esto, entre las numerosas metodologías que existen, he escogido la metodología ágil SCRUM, ya que es una forma de organizar fácilmente las tareas que hay que sacar adelante.
Esta metodología se basa en la creación de tareas pequeñas, normalmente etiquetadas por categorías (como pueden ser diseño, implementación, redacción de la memoria, investigación de tecnologías, etc.), las cuales se agrupan en tres estados ("por hacer", "en progreso" y "hecho").
Para tener una mejor visión de estas categorías, se suele utilizar un tablero donde se muestran los 3 estados, mencionados anteriormente, representados por 3 columnas donde se van colocando las diferentes tareas. Todo esto permite poder analizar los avances conseguidos o los objetivos pendientes, ayudando a mantener la motivación y no ver las tareas como compromisos titánicos, difíciles de conseguir o sin saber cómo abordarlos.

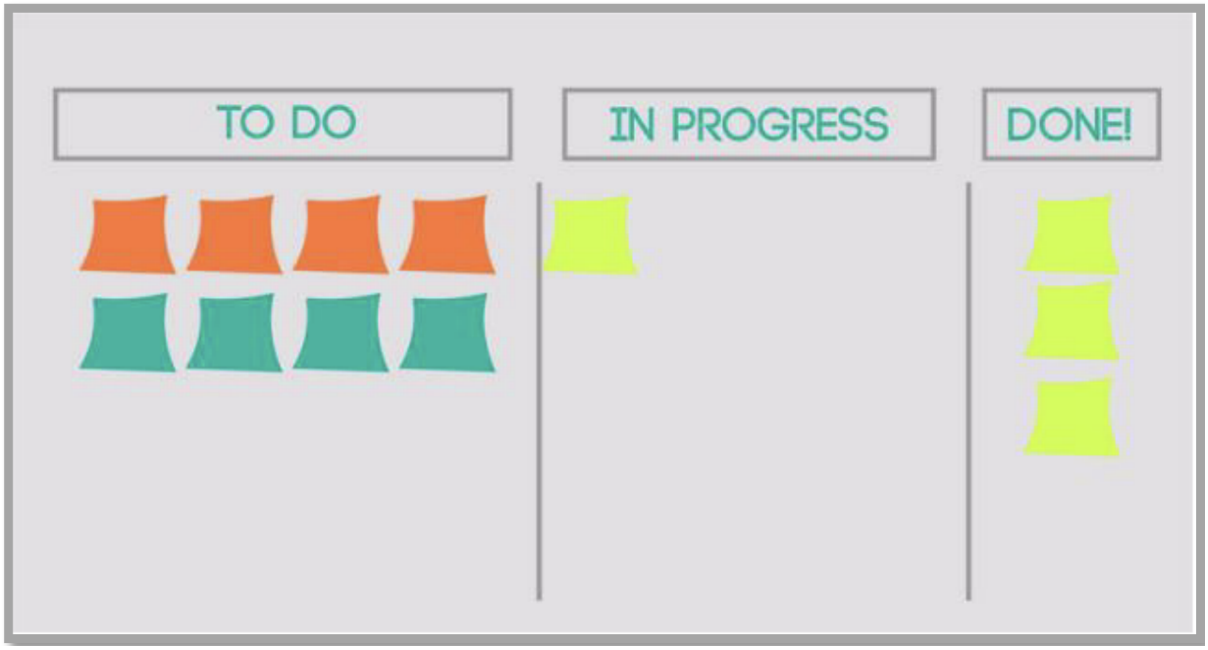


Figura 6 - Metodología ágil SCRUM

Cabe mencionar, además, que estas tareas se van colocando en función de diferentes hitos, también llamados "sprints", los cuales permiten fijar un límite de tiempo para un conjunto de ellas. En mi caso, he solido utilizar sprints semanales, pero de vez en cuando he tenido retrasos, los cuales me han obligado a reestructurar los sprints.

Existen diferentes herramientas que ayudan a la construcción del tablero, como puede ser una simple pizarra con post-its o programas dedicados a ello como Trello. En mi caso, decidí utilizar la herramienta que proporciona GitHub para este fin, ya que tengo en dicha plataforma el proyecto subido y me es más cómodo trabajar desde ahí.
Es cierto que github no tiene tantas opciones como es Trello, pero cumple satisfactoriamente con mis necesidades.

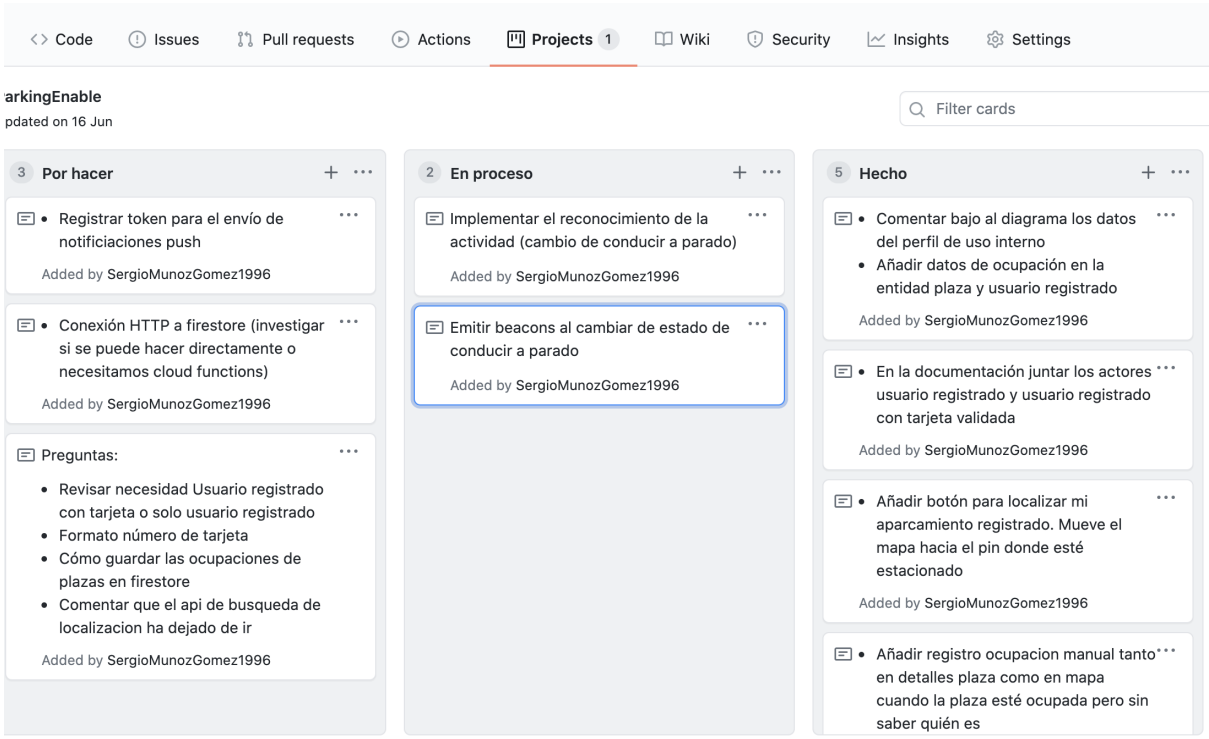


Figura 7 - Tablero tareas SCRUM Github

4. Objetivos

En este apartado voy a tratar los diferentes objetivos que se han planteado para el desarrollo del proyecto.
Partiendo de la reunión inicial del proyecto, hemos sacado una serie de objetivos:

- Gestión de usuarios. Con este objetivo se quiere tener un control de los datos del usuario que hará uso de la aplicación para, por ejemplo, validar su tarjeta de minusválido. Dependiendo del estado del usuario en la aplicación, las funciones de esta variarán. Estos estados son:
    - Persona con la aplicación y con tarjeta de minusválido verificada.
    - Persona con la aplicación, pero con la tarjeta de minusválido sin verificar.
    - Persona con la aplicación, pero sin tener tarjeta de minusválido.

- Comunicación con la baliza. Con este objetivo se quiere plasmar la correcta conexión entre el móvil y la baliza, la cual depende del estado del usuario en la aplicación.
- Registro manual del aparcamiento en la plaza tiempo después de haber aparcado, por si el registro automático hubiera fallado o no esté disponible.

5. Análisis y especificación

En este apartado voy a centrarme en los diferentes aspectos que aborda el proyecto para analizarlos y así dividirlos en pequeños puntos que tratar. Hacer esto me permite pasar la idea global que tengo del proyecto en subtareas que me ayudarán a visualizar los diferentes componentes que tengo que tratar.
Para ello vamos a diferenciar diferentes requerimientos que serán los que definan las futuras funcionalidades que tendrá la aplicación. Estos requerimientos parten de análisis de los diferentes actores que van a interactuar con el sistema, los requisitos de usuario que definen lo que los actores podrán hacer, los requisitos funcionales que definen lo que el sistema puede hacer y requisitos no funcionales que son una descripción de una propiedad que debe tener el sistema o una restricción que debe respetar.

5.1. Actores

- **Usuario no registrado**. El actor usuario no registrado, también referido como usuario invitado, representa a aquel usuario que utiliza la aplicación, pero sin haber realizado ningún tipo de registro en la misma. Sus funcionalidades son la capacidad de registrarse para tener un perfil propio y la de poder consultar plazas de aparcamientos para minusválidos en un mapa.

- **Usuario registrado**. El actor usuario registrado con tarjeta de movilidad reducida representa a aquel usuario que ha realizado el proceso completo de registro de un perfil en la aplicación, es decir, junto a sus datos personales ha adjuntado la información de su tarjeta de movilidad reducida. Su funcionalidad es completa ya que podrá gestionar su perfil, hacer uso del registro automático de su aparcamiento y registrarlo manualmente, además de la funcionalidad básica del usuario no registrado.

5.2. Requisitos de usuario

5.2.1. Requisitos de usuario del actor usuario no registrado

- Gestión del mapa
    - Encontrar la ubicación del usuario.
    - Seleccionar el estilo del mapa.
    - Buscar ubicación a través de un buscador.

- Gestión aparcamiento
    - Buscar aparcamiento a través de un mapa.
    - Ver detalles de aparcamiento.
    - Navegar hacia el aparcamiento.
    - Notificar problema aparcamiento.
    - Sugerir nueva plaza de aparcamiento.

- Gestión acceso al perfil
    - Registrarse en la aplicación.
    - Identificarse en la aplicación.

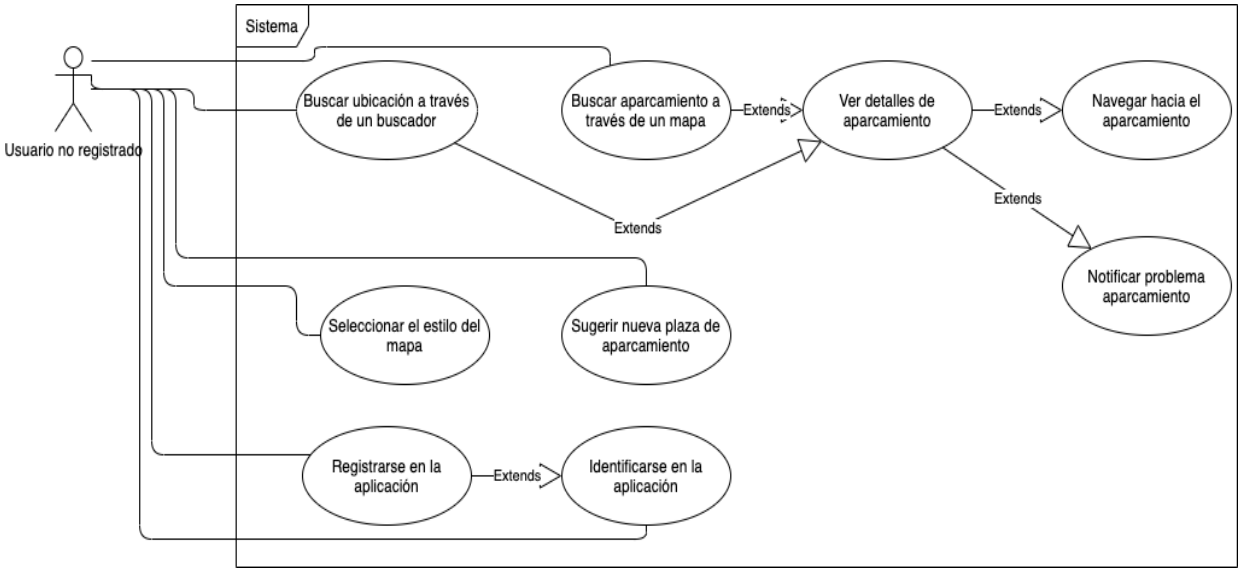


Figura 8 - Diagrama de casos de uso del actor Usuario no registrado

5.2.2. Requisitos de usuario del actor usuario registrado

- Permitir registro automático del aparcamiento
- Gestión del aparcamiento
    - Registrar manualmente su aparcamiento.
    - Consultar ubicación del aparcamiento del usuario.

- Gestión del perfil (CRUD)
    - Consultar perfil.
    - Modificar perfil (los datos que se pueden modificar dependen del estado de verificación del perfil).
    - Desidentificarse.
    - Borrar perfil.

- Votar estado del aparcamiento.

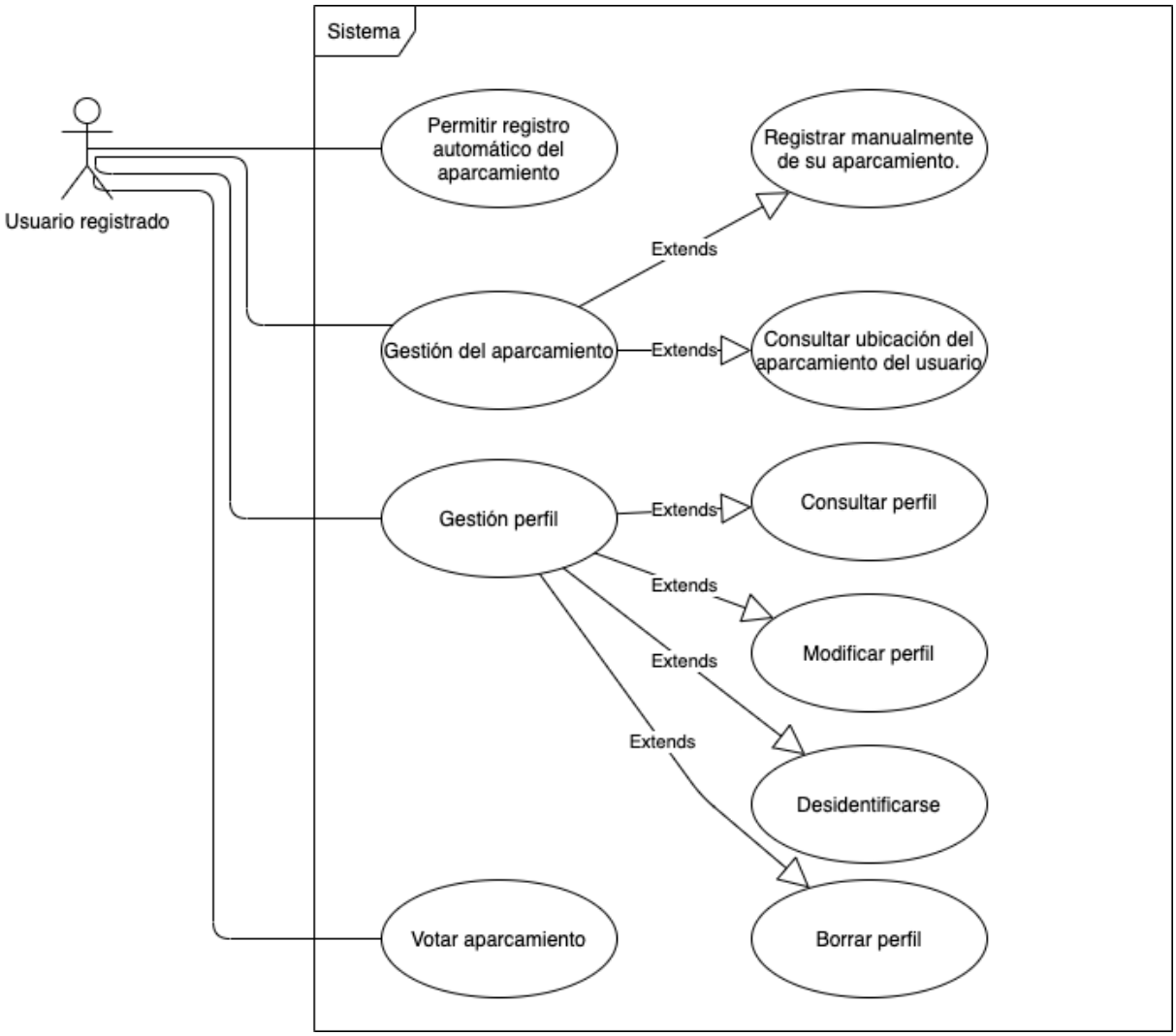


Figura 9 - Diagrama de casos de uso del actor Usuario registrado

5.3. Requisitos funcionales

Identificador	RF-1
Nombre	Pedir permisos ubicación
Descripción	El sistema pedirá acceso a la ubicación del dispositivo para poder hacer uso de aquella funcionalidad que la requiera
Prioridad	Alta

Identificador	RF-2
Nombre	Mostrar mapa funcional
Descripción	El sistema mostrará un mapa donde aparecerán todos los aspectos relacionados con él.
Prioridad	Alta

Identificador	RF-3
Nombre	Mostrar ubicación del usuario
Descripción	El sistema mostrará en el mapa la ubicación actual del usuario
Prioridad	Media

Identificador	RF-4
Nombre	Mostrar direcciones de búsqueda
Descripción	El sistema mostrará sugerencias de direcciones parecidas a la buscada
Prioridad	Media

Identificador	RF-5
Nombre	Mostrar estilos del mapa
Descripción	El sistema mostrará el mapa en el estilo previamente seleccionado por el usuario.
Prioridad	Baja

Identificador	RF-6
Nombre	Mostrar marcadores representando aparcamientos
Descripción	El sistema mostrará diferentes marcadores para localizar los diferentes aparcamientos disponibles y el estado de ocupación del mismo en diferentes colores, siendo el verde para indicar su no ocupación y el rojo para indicar que está ocupado
Prioridad	Alta

Identificador	RF-7
Nombre	Abrir maps para mostrar ruta hacia la plaza seleccionada
Descripción	El sistema abrirá la aplicación de mapas Google Maps para mostrar una ruta hacia la plaza seleccionada
Prioridad	Alta

Identificador	RF-8
Nombre	Mostrar detalles de plaza de aparcamiento
Descripción	El sistema mostrará una pantalla donde se detallarán los datos de la plaza de aparcamiento consultada previamente por el usuario.
Prioridad	Alta

Identificador	RF-9
Nombre	Guardar votación de la plaza
Descripción	El sistema guardará la votación que el usuario haya dado respecto a la plaza en función de su estado o su accesibilidad.
Prioridad	Media

Identificador	RF-10
Nombre	Mostrar formulario de registro
Descripción	El sistema mostrará un formulario pidiendo al usuario los datos necesarios para realizar el proceso de registro y así pasar a ser un usuario registrado.
Prioridad	Alta

Identificador	RF-11
Nombre	Guardar datos de registro
Descripción	El sistema guardará los datos del usuario que haya introducido en el formulario de registro y así completar este proceso.
Prioridad	Alta

Identificador	RF-12
Nombre	Mostrar formulario de login
Descripción	El sistema mostrará un formulario para que el usuario introduzca sus datos necesarios para realizar el proceso de login
Prioridad	Alta

Identificador	RF-13
Nombre	Comprobar datos de login
Descripción	El sistema comprobará los datos que el usuario en el formulario de login para validar que representan a un usuario previamente registrado.
Prioridad	Alta

Identificador	RF-14
Nombre	Mostrar perfil
Descripción	El sistema mostrará los datos del perfil del usuario para que éste pueda consultarlos.
Prioridad	Alta

Identificador	RF-15
Nombre	Desidentificar al usuario
Descripción	El sistema borrará la referencia de los datos de acceso del usuario en la aplicación para que se pueda volver a realizar el proceso de identificación.
Prioridad	Alta

Identificador	RF-16
Nombre	Borrar información del usuario
Descripción	El sistema, tras una confirmación del usuario, borrará del servidor los datos de perfil y los añadirá al histórico de usuarios borrados.
Prioridad	Alta

Identificador	RF-17
Nombre	Activar bluetooth
Descripción	El sistema notificará y activará el bluetooth, tras previa autorización del usuario
Prioridad	Media

Identificador	RF-18
Nombre	Permitir acceso a reconocimiento de actividad del usuario
Descripción	El sistema pedirá acceso, si es necesario, para recoger la actividad del usuario.
Prioridad	Alta

Identificador	RF-19
Nombre	Detectar cambio de actividad del usuario
Descripción	El sistema recogerá los cambios de actividad que se hayan fijado, concretamente cuando el usuario deje de conducir.
Prioridad	Alta

Identificador	RF-20
Nombre	Emitir como beacons
Descripción	El sistema, si tiene el bluetooth activado, emitirá señal bluetooth con el identificador del usuario como si fuera un beacon
Prioridad	Alta

Identificador	RF-21
Nombre	Notificar al usuario de que se ha guardado su aparcamiento
Descripción	El sistema mostrará una notificación al usuario indicando que el aparcamiento se ha registrado correctamente, de esta manera se completa la funcionalidad de registro automático del aparcamiento.
Prioridad	Alta

Identificador	RF-22
Nombre	Mostrar opción para registro del aparcamiento manual
Descripción	El sistema mostrará una opción para que el usuario pueda registrar su ocupación en una plaza, siempre y cuando la plaza esté ocupada pero sin saber por quien.
Prioridad	Alta

Identificador	RF-23
Nombre	Guardar la información del usuario en la plaza ocupada seleccionada
Descripción	El sistema guardará el identificador del usuario indicando que la plaza seleccionada está ocupada por el usuario.
Prioridad	Alta

Identificador	RF-24
Nombre	Mostrar ubicación del aparcamiento del usuario
Descripción	El sistema mostrará, siempre y cuando el usuario esté ocupando una plaza, el marcador de la plaza de color azul y además, tendrá una opción para localizarlo rápidamente.
Prioridad	Alta

Identificador	RF-25
Nombre	Aparcamiento único
Descripción	El actor Usuario Registrado con tarjeta de movilidad reducida solo podrá registrar un aparcamiento a la vez y siempre que la plaza esté ocupada. Si quiere registrar otro aparcamiento deberá borrar el registro del aparcamiento anterior.
Prioridad	Alta

Identificador	RF-26
Nombre	Valoración personal
Descripción	Solo el actor Usuario Registrado podrá valorar plazas de aparcamiento. Lo hará proporcionando como mucho una única valoración, es decir, el actor sólo podrá valorar una plaza positiva o negativamente, si cambia su valoración se actualizará el estado de la valoración de ese usuario a esa plaza.
Prioridad	Media

Identificador	RF-27
Nombre	Modificación condicionada
Descripción	El actor Usuario Registrado solo podrá modificar los atributos del nombre, apellidos, número de tarjeta de movilidad reducida, fecha de caducidad y foto de la tarjeta mientras el perfil no esté verificado, una vez verificado solo podrá modificar el resto de los atributos.
Prioridad	Alta

5.4. Requisitos no funcionales

Los requisitos no funcionales son una descripción de una propiedad que debe tener el sistema o una restricción que debe respetar. Este tipo de requisitos también describe servicios o características de rendimiento del sistema. Estos requisitos describen características importantes del sistema: disponibilidad, usabilidad, seguridad, rendimiento, idioma, moneda, compatibilidad…

Identificador	RNF-001
Nombre	Almacenamiento de contraseñas
Descripción	Las contraseñas de los usuarios se almacenarán utilizando una función HASH (Seguridad).
Prioridad	Alta

Identificador	RNF-002
Nombre	Seguridad en las conexiones
Descripción	Toda la comunicación privada se llevará a cabo utilizando el protocolo seguro HTTPS (Seguridad).
Prioridad	Alta

Identificador	RNF-003
Nombre	Ley Orgánica de Protección de Datos
Descripción	La aplicación cumplirá todas las condiciones de las Ley Orgánica de Protección de Datos (Seguridad).
Prioridad	Alta

Identificador	RNF-004
Nombre	Acceso login/password
Descripción	Todos los usuarios registrados accederán al sistema mediante un par login/password que se asociará a cada perfil, concretamente usará su número de tarjeta junto con su contraseña. (Seguridad)
Prioridad	Alta

Identificador	RNF-005
Nombre	Modificación privada y condicionada
Descripción	Un usuario podrá modificar únicamente los datos asociados a su información de registro, es decir, hay datos generados por el sistema los cuales no podrá modificar (Integridad).
Prioridad	Alta

Identificador	RNF-006
Nombre	Transacciones y atomicidad en los datos
Descripción	Las operaciones se definirán como transacciones y se garantizará su atomicidad (Fiabilidad).
Prioridad	Alta

Identificador	RNF-007
Nombre	Disponibilidad del sistema
Descripción	El sistema estará disponible 24 horas al día, los 7 días de la semana, excepto en puntuales momentos de mantenimiento (Disponibilidad).
Prioridad	Media

Identificador	RNF-008
Nombre	Datos actualizados
Descripción	El servidor deberá notificar a la aplicación cualquier cambio en la base de datos para que disponga de los datos actualizados en tiempo real. (Disponibilidad)
Prioridad	Alta

Identificador	RNF-009
Nombre	Interfaz intuitiva
Descripción	La aplicación contará con una interfaz sencilla e intuitiva para hacer más rápida la experiencia de usuario, utilizando metáforas en menús y botones. Además, contará con un diseño llamativo para captar la atención de los usuarios. (Usabilidad).
Prioridad	Media

Identificador	RNF-010
Nombre	Funcionalidad exclusiva al registro
Descripción	Solo el actor Usuario Registrado con tarjeta de movilidad reducida podrá utilizar las características del registro de aparcamiento.
Prioridad	Alta

Identificador	RNF-011
Nombre	Acceso privado
Descripción	Al actor Usuario Registrado se le permitirá acceder y modificar solo la información de su perfil.
Prioridad	Alta

Identificador	RNF-012
Nombre	Actualización tarjeta caducada
Descripción	El actor Usuario Registrado, aunque tenga el perfil verificado, si la tarjeta de movilidad reducida se ha caducado podrá actualizar la fecha de caducidad y actualizar la foto de la nueva tarjeta.
Prioridad	Alta

Identificador	RNF-013
Nombre	Número de tarjeta único
Descripción	El actor Usuario Registrado solo podrá modificar el número de la tarjeta de movilidad reducida por uno que no esté registrado, para así evitar suplantaciones de identidad.
Prioridad	Alta

Identificador	RNF-014
Nombre	Longitud contraseña
Descripción	Tanto en el proceso de registro, como en el de edición del perfil, el actor Usuario Registrado deberá introducir una contraseña con 6 caracteres o más.
Prioridad	Media

Identificador	RNF-015
Nombre	Validación de contraseña
Descripción	Tanto en el proceso de registro, como en el de edición del perfil, el actor Usuario Registrado deberá validar que los campos de la contraseña no se queden vacíos y que la contraseña nueva coincida con la contraseña revisada, para confirmar la validez de los datos.
Prioridad	Alta

Identificador	RNF-016
Nombre	Indicación datos obligatorios
Descripción	En todos los formularios, el sistema deberá indicar al usuario aquellos datos que son obligatorios e informar de aquellos que falten por rellenar.
Prioridad	Media

Identificador	RNF-017
Nombre	Acceso concedido para registro automático aparcamiento
Descripción	El Usuario Registrado deberá mantener el permiso de reconocimiento de actividad otorgado y el bluetooth activado, para mantener la opción del registro manual del aparcamiento.
Prioridad	Media

6. Diseño

En este apartado se detallarán todos los componentes y arquitecturas lógicas y físicas que constituyen el diseño de la aplicación móvil.

6.1. Arquitectura lógica

La arquitectura lógica define los elementos lógicos que forman la aplicación y cómo se relacionan entre ellos.
La explicación de este apartado está desarrollada desde una visión más global, para entender las diferentes características del sistema de una manera conjunta y argumentar el porqué de cada una, hasta una visión más detallada donde se contemplan los componentes que forman cada parte.

Desde un punto de vista a alto nivel, nuestro sistema se puede dividir en 3 capas:

- La capa de presentación. Esta capa está formada por los componentes que describen la interfaz de usuario. Son diferentes elementos que nos proporcionan una vista intuitiva, atractiva y funcional para llevar a cabo las acciones que detalla la capa de negocio.
- La capa de negocio. Esta capa contiene toda la lógica de la aplicación. Realiza, a través de métodos, todas aquellas funciones que nos permiten comunicar esta capa con la capa de presentación, entre componentes de esta misma capa y con el sistema de datos.
- La capa de datos. Esta capa es la encargada de almacenar y gestionar la información (creación, acceso, actualización y borrado).

El motivo de dividir el sistema en estas 3 capas es porque nos permite agrupar elementos similares y separarlos de aquellos que no cumplen funciones parecidas. Esto permite mejorar la comunicación entre elementos y, a la vez, nos da mayor facilidad a la hora de aplicar cambios en alguna de estas capas, ya que son independientes.

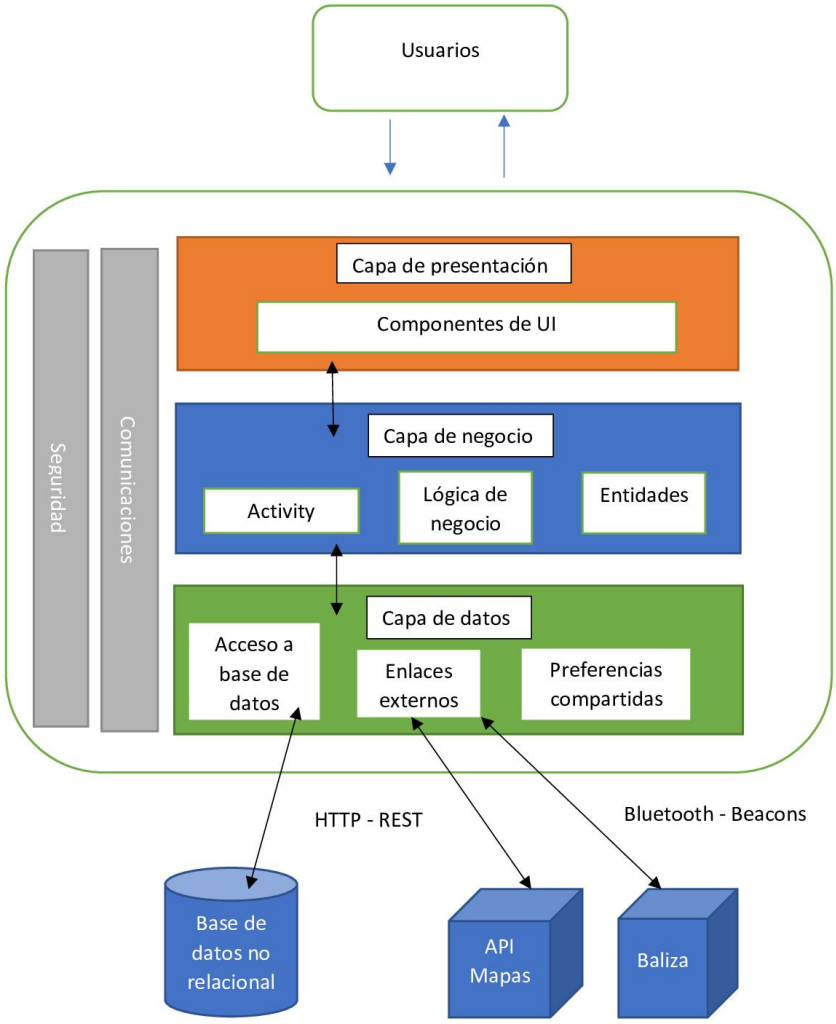


Figura 10 - Arquitectura lógica

6.2. Arquitectura física

Para la arquitectura física se ha utilizado un modelo en 3 capas siguiendo la misma filosofía que en el diseño lógico. En este caso, es un esquema cliente-servidor-base de datos. La diferencia que hay es que, en vez de disponer de servidores diferentes para la aplicación y la base de datos, todas las conexiones se hacen a los servicios de Firebase el cual dispone de múltiples servidores distribuidos por módulos que representan diversas funcionalidades y se comunican entre sí.

Para este proyecto se ha decidido implementar la arquitectura física de una manera muy sencilla. Se utilizan los servidores de Google (Firebase) como "software as a service" para todo el tema de conexiones, seguridad de las mismas y balanceo de carga. Por este motivo, para nuestro desarrollo toda esta parte forma una caja negra que nos da un servicio a través de unas reglas que nos permiten su uso, por lo que nosotros nos tenemos que ocupar de la gestión de las respuestas que nos proporcionan los diferentes módulos de Firebase que hemos usado, incluyendo el servicio de base de datos en tiempo real.

A continuación, se muestra el diagrama de despliegue físico del sistema. En él se explica la disposición que hay entre los componentes y cómo se comunican.

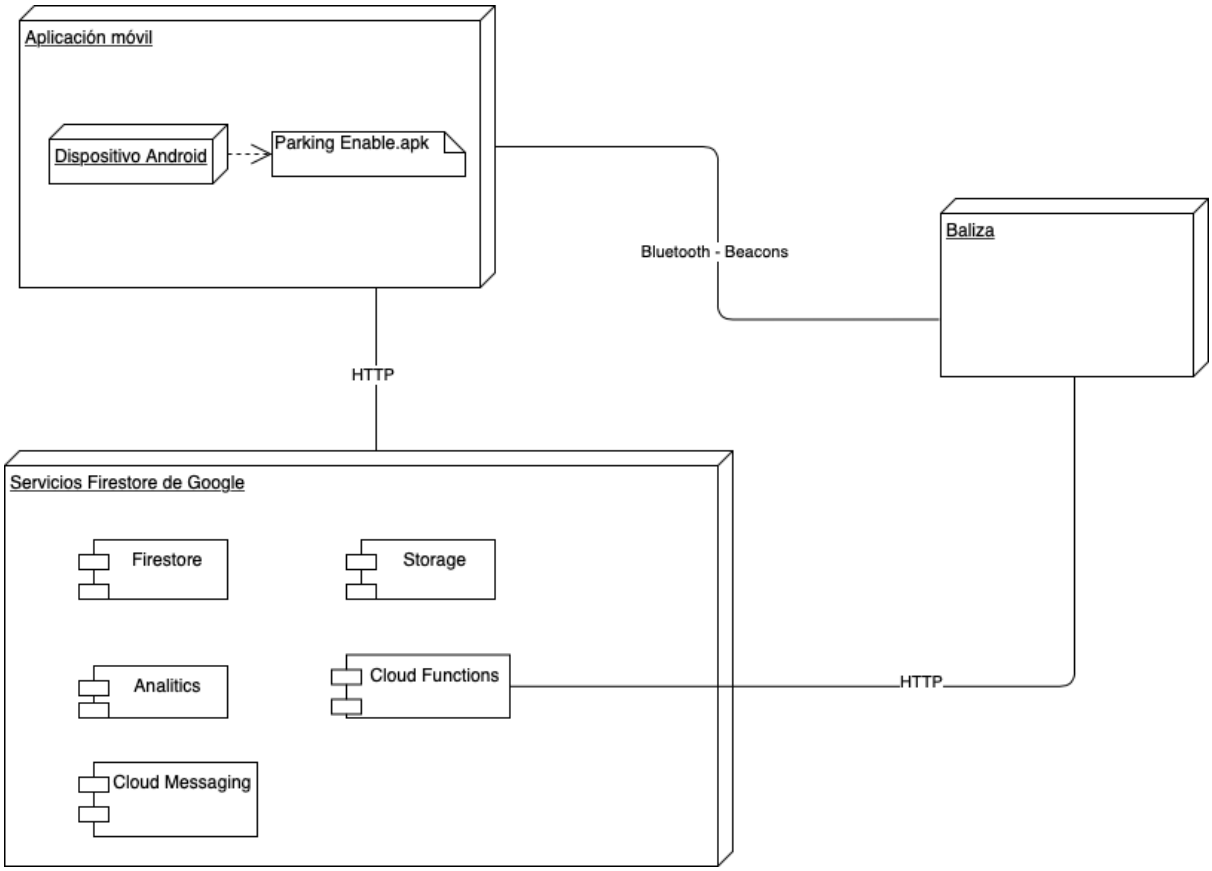


Figura 11 - Arquitectura física

6.3. Modelo Entidad - Relación (E-R)

El modelo Entidad-Relación ilustra las entidades y los tipos de relaciones que pueden darse entre las entidades.

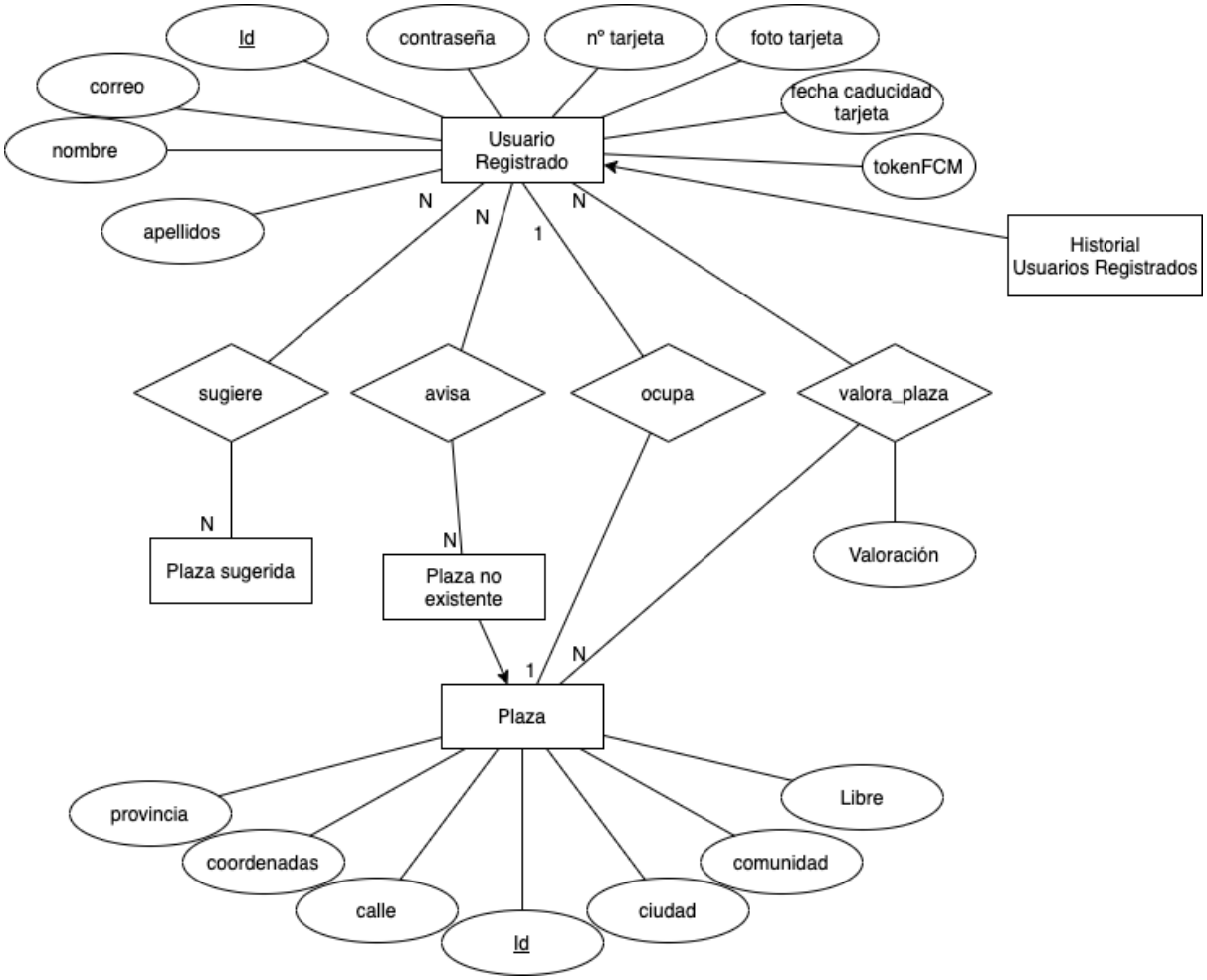


Figura 12 - Diagrama Entidad/Relación

Como vemos en el diagrama, tenemos dos entidades clave en el diseño de la base de datos, estas son usuario registrado y plaza, ya que representan los datos en los que se basa la aplicación.

- La entidad Usuario registrado tiene los atributos descritos en el diagrama, pero además, dispone de otros que se almacenan para facilitar la administración de las instancias de dicha entidad y la posibilidad de utilizarlos, en un futuro, para añadir más funcionalidades a la aplicación. Estos atributos extra son:
    - Correo verificado.
    - Fecha de creación.
    - Fecha última modificación.
    - Fecha revisión.
    - Perfil revisado. Sirve para indicar que un administrador ha verificado los datos del perfil.

    - Perfil suspendido. Sirve para indicar que un administrador ha suspendido la cuenta por fraude o uso indebido.
    - Código para el cambio de contraseña.
    - Fecha de caducidad del código de cambio de contraseña.

  Ligada a esta entidad, tenemos la entidad Historial usuario registrados, la cual hereda de Usuarios registrados, ya que tiene los mismos atributos. Esta entidad representa el registro de aquellos perfiles que hayan sido borrados.

- La entidad Plaza hace referencia a las plazas de aparcamientos que se hayan confirmado como plazas de minusválidos existentes. De esta entidad surgen otras como Plaza sugerida o Plaza no existente que nos sirven para especializar el estado de una plaza.
  Cabe mencionar que una plaza puede ser ocupada, pero sin saber por quién, por lo que se podrá actualizar el estado de la misma pero sin la necesidad de asociarlo a alguien.
- Estas entidades se relacionan para indicar que un usuario ha valorado, ocupado, sugerido o avisado del estado de una plaza.

Además de todo esto, cabe mencionar que este diseño está orientado para ser utilizado en una base de datos no relacional, por lo que la estructura cambia ligeramente para poder ser implementada. Concretamente, el diseño de la base de datos estará destinada a ser una base de datos no relacional basada en documentos, por lo que seguirá un esquema en forma de árbol. Partimos de las entidades, siendo los nodos que salen del nodo raíz, y a partir de ellos es donde se organizan los datos. Se ha elegido este tipo de base de datos porque los datos no tienen por qué cambiar en exceso y permite consultas mucho más rápidas.
Debido a esta estructura, he decidido replicar algunos datos para mantener las referencias entre nodos, por ejemplo, en los documentos del nodo usuario tenemos un atributo que representa la plaza ocupada, y de la misma manera ocurre en los documentos del nodo plazas donde tenemos un atributo que representa por quién está ocupada. Esto se ha diseñado así para tener un mayor y rápido acceso a los datos en función de cuáles se necesiten, debido a que al ser una base de datos no relacional tenemos que gestionar de manera manual dichas relaciones.

6.4. Diseño de la interfaz

Este apartado proporciona una visión aproximada sobre cómo serán las interfaces de usuario del sistema. Por lo tanto, la aplicación final puede sufrir diferentes cambios estéticos.
Para la creación de bocetos se ha utilizado una aplicación web denominada “Ninja mok”.
La cual nos permite construir de manera rápida y sencilla interfaces orientadas a móviles muy básicas, pero con numerosos elementos.

6.4.1. Pantalla principal

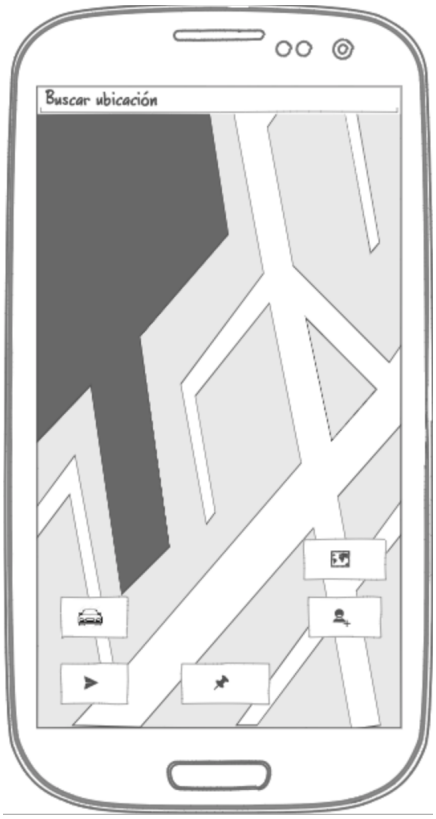


Figura 13 - Boceto pantalla principal

En cuanto a esta pantalla decir que es la que define la aplicación, es decir, la pantalla que posee casi todas las funcionalidades esenciales para el usuario.

En la pantalla principal tendremos un mapa de donde podremos ver la ubicación actual del dispositivo que estemos utilizando y las marcas (chinchetas o markers) que nos indicarán donde podremos encontrar plazas de aparcamiento para personas con problemas de movilidad.

De la misma forma, tendremos ubicado en la zona superior un buscador, donde pulsando se nos abrirá un teclado para introducir un lugar, una calle... Además, este buscador tendrá una lista de sugerencias de lugares y calles relacionados con lo que vamos escribiendo en la barra.

En cuanto a la barra de búsqueda procuraremos que sea una búsqueda guiada con la cual podamos mostrar al usuario una lista de sugerencias mientras escribe de forma que podamos facilitar algo el encontrar el lugar al que se dirige.

Por otro lado, el botón que centra la pantalla es muy usado en todas las aplicaciones que contienen un componente de mapas, ya que podremos movernos libremente por el mapa y sea fácil volver mediante una transición al punto en el que se encuentra el usuario.

Tenemos otro botón con el que vamos a poder implementar una funcionalidad más interactiva por parte de los usuarios de esta aplicación. Ya que no tenemos una base de datos pública a la que acudir para almacenar todas las plazas de aparcamiento para personas discapacitadas con este botón lo que buscaremos es que los interesados puedan mandarnos al sistema Firebase su ubicación indicándonos que ahí existe una plaza de aparcamiento que no tenemos registrada o no tenemos reflejada correctamente, de forma que la persona encargada de la aplicación (administrador) o un posterior sistema de procesado de datos pueda crear nuevas plazas de aparcamiento en la base de datos de EnableParking.

También podremos ver un botón/menú flotante que nos permitirá cambiar la vista del mapa entre el estándar y la vista por satélite, que es un mapa con fotografías reales. Esto puede ayudar cuando una zona nos es familiar de forma que de un vistazo sepamos en la zona en la que nos encontramos.

En cuanto a los marker o chinchetas que marcan la ubicación de las plazas de aparcamiento serán pulsables, de forma que nos lleven a una pantalla personalizada de cada una de las plazas de aparcamiento donde podremos ver mediante una interfaz sencilla y con buen tamaño a qué plaza nos referimos y datos sobre ella.

Todo lo descrito anteriormente eran aportaciones que el compañero que empezó la aplicación ya tenía en funcionamiento.

A partir de aquí, mi aportación fue la inclusión de un botón para acceder a la pantalla de login o, si ya estamos logeados en el sistema, nos llevará a la pantalla de nuestro perfil para consultar nuestros datos y gestionarlos.

A parte de esto, contamos con un botón que nos permite registrar manualmente nuestra ocupación en una plaza (que esté ocupada y sin un usuario registrado como su ocupante). También contamos con otro botón que nos permite encontrar la plaza en donde hemos aparcado, siempre y cuando tengamos una plaza ocupada con nuestro perfil.

6.4.2. Pantalla detalles de aparcamiento

Figura 14 - Boceto pantalla detalles de aparcamiento

Primeramente, la calle y el número sobre el que se encuentra, como aproximación y punto de referencia. Después tendremos tres botones grandes e interactivos. Dos de ellos para puntuar la plaza de aparcamiento y un tercero para unas de las funcionalidades más importantes de la aplicación: navegar hasta la plaza indicada. Para esto hemos hecho uso de la aplicación de Google Maps que poseemos todos en nuestro teléfono móvil, de forma que al pulsar el botón se abra y nos guíe.
Para finalizar, además de mostrar la puntuación de "me gusta" / "no me gusta" de la plaza de aparcamiento, tendremos un botón para volver a la pantalla principal.

Lo anteriormente descrito forma parte de la aportación que hizo el compañero que inició la aplicación. A partir de aquí, lo que hice fue una reestructuración de la interfaz para darle un aspecto más intuitivo, teniendo más tamaño y estando más arriba aquellos botones que sean más importantes o que se vayan a usar más. Además, he añadido el botón del registro de aparcamiento manual para poder acceder a esta funcionalidad, siempre y cuando la plaza esté en un estado que lo permita.

6.4.3. Pantalla de registro de usuario

Figura 15 - Boceto pantalla de registro de usuario

Esta pantalla representa el formulario que tiene que seguir cualquier usuario para registrarse en la aplicación y así tener un perfil que lo identifique, además de poder acceder a mayor funcionalidad.

Este formulario consta de diferentes entradas de texto para la información necesaria del perfil, tales como el correo de contacto, nombre, apellidos, contraseña, número de la tarjeta de movilidad reducida, una foto de la misma que servirá para validar dicho perfil, fecha de caducidad, etc. Cabe destacar, que el formulario está diseñado para que sea necesario rellenar todos los campos descritos para empezar con la creación del perfil.

6.4.4. Pantalla login

Figura 16 - Boceto pantalla login

Esta pantalla hace referencia al proceso de login de un usuario ya registrado. Por eso, contamos con un formulario en el que nos pide el número de la tarjeta de movilidad reducida y la contraseña.
Además, esta pantalla sirve de puente entre la principal y la de registro para permitir al usuario crearse un perfil si así lo desea.

6.4.5. Pantalla perfil

Figura 17 - Boceto pantalla perfil

Esta pantalla hace referencia al perfil del usuario registrado, ya que se muestran todos los datos que forman su perfil, excepto algunos datos guardados de ámbito administrativo, y da acceso a otras funcionalidades como cerrar sesión, el borrado del perfil o la edición de este. Respecto a la edición del perfil, la pantalla es muy similar a esta, solo que cambia en función del estado del usuario. Por ejemplo, hay ciertos datos que solo son editables si el perfil no ha sido verificado o si la fecha de caducidad de la tarjeta aún no ha llegado.

Por último, quiero mencionar que en esta pantalla contamos con la opción de activar o desactivar el registro automático del aparcamiento. Para activar esta opción deberemos tener concedido el permiso de reconocimiento de la actividad y tener encendido el bluetooth, ya que son aspectos necesarios para la comunicación con la baliza.

6.5. Diagramas de secuencia

En este apartado se describen las interacciones que hay entre los componentes del sistema a lo largo del tiempo, a través de diagramas de secuencia.

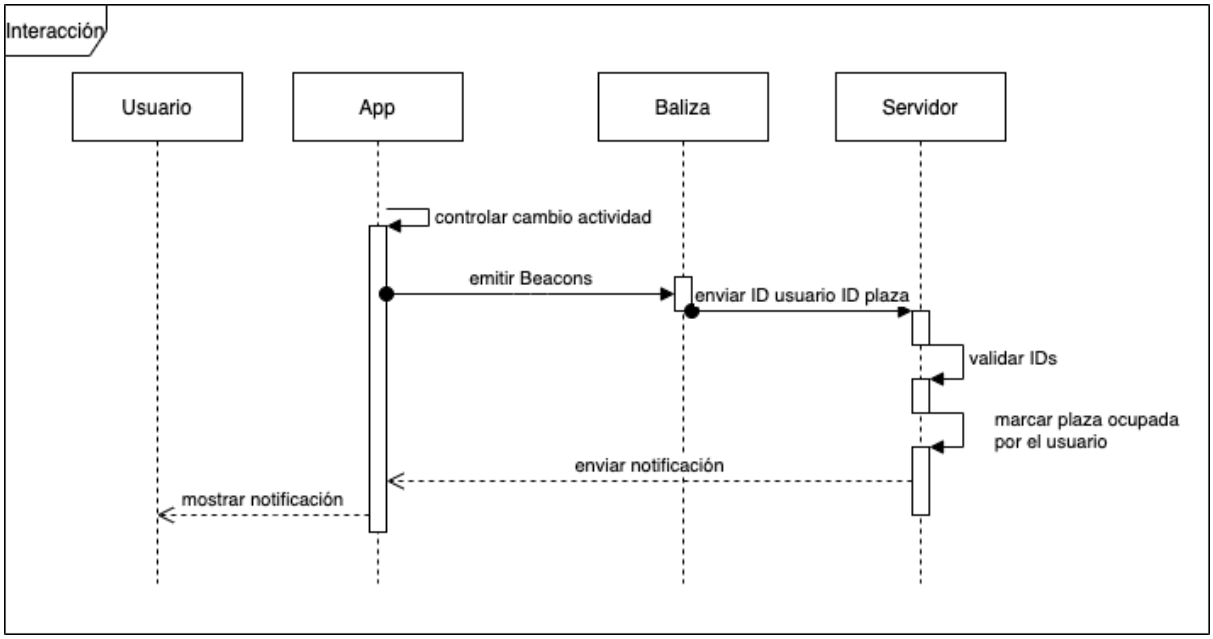


Figura 18 - Diagrama de secuencia del registro automático de la ocupación

El diagrama de secuencia nos ilustra cómo funciona el proceso de registro de aparcamiento automático, siendo ésta una de las funcionalidades más importantes de la aplicación.
Este proceso parte del estado en el que el móvil sea apto para ejecutar este procedimiento, es decir, que el usuario esté registrado, tenga la opción activada y todos los recursos (reconocimiento de la actividad y bluetooth) activados.
Una vez, teniendo todo listo, el proceso empieza monitorizando la actividad que tenemos con nuestro dispositivo android. Cuando se reconozca que ha habido la transición de actividad de dejar de conducir, lo que representa aparcar el coche, empezará a emitir una señal bluetooth en forma de Beacon donde llevará consigo el ID del usuario. La baliza estará escaneando beacons y cuando reciba uno, recogerá el ID del usuario procedente del Beacon y junto con la ID de la plaza a la que está asociada mandará una petición HTTP al servidor con dichos datos. Cuando el servidor recibe la petición HTTP recogerá y validará los IDs recibidos para actualizar la base de datos indicando que el usuario indicado está ocupando dicha plaza. En ese momento el servidor enviará una notificación al móvil del usuario para indicarle que el proceso de aparcamiento se ha completado satisfactoriamente.

6.6. Guía de estilos

En este apartado vamos a ver la guía de estilos que se utilizarán en el proyecto.
Esto significa que todo el diseño de la aplicación girará alrededor del estilo que hemos decidido con anterioridad. En cuanto a la gama de colores e iconos nos hemos basado en los principios de Material Design.

Por una parte, los colores que hemos escogido han sido unos colores fáciles de complementar y que pensamos para que por el uso de la aplicación no cansen además de una paleta amplia para cubrir las necesidades que teníamos. Para generar la paleta hemos usado una aplicación web denominada Coolors [23] y este es el resultado:

Figura 19 - Guía de colores

Por otra parte, los iconos han sido extraídos de varias fuentes. La principal ha sido Android Studio, que pone a nuestra disposición un librería amplia y versátil, pero en otros casos donde pensamos que queríamos algo más atractivo visualmente hemos acudido a librerías web donde podemos usar varias de las creaciones que la comunidad pone a nuestra disposición de manera desinteresada como por ejemplo flaticon, Noun Project y Map Icons, entre otras.

Cabe destacar, que la guía de estilos fue elegida por el compañero que empezó la aplicación, en mi caso, simplemente he seguido con ella para dar homogeneidad a las interfaces.

7. Implementación

En este apartado se describen los detalles de implementación de la aplicación móvil.
En primer lugar, se detallarán los aspectos relacionados con el servidor, seguido de la comunicación las balizas. En segundo lugar, las tecnologías utilizadas. En tercer lugar, las herramientas empleadas. En cuarto lugar, los detalles de implementación y por último los detalles de despliegue.

7.1. Servidor

Como se ha descrito en diferentes puntos de la documentación, se ha configurado un servidor proporcionado por Google como servicio (SAAS), llamado Firebase, al cual se conectará nuestra aplicación móvil para todas aquellas funciones remotas.
La motivación para la elección de esta solución se basa en la innovación que trae este servicio. Aunque tenga pocos años de uso, Google se ha preocupado de construir una plataforma que nos proporcione desde un mismo lugar numerosas y variadas funcionalidades para que aplicaciones web, IOS y, en nuestro caso, Android las puedan usar.
La principal ventaja de Firebase es que sus diferentes módulos están diseñados para una fácil relación y comunicación entre ellos, independientemente del tipo de aplicación. Además, permite implementar todos los módulos de una manera rápida y sencilla, solo hay que saber cómo trabajar con los diferentes módulos, y tampoco hay que preocuparse de administrar la infraestructura.
Otra característica es que contamos con el respaldo de Google, por lo que tenemos la confianza de manejar un entorno seguro donde nuestros datos e infraestructura son siempre accesibles y escalables. Además de contar con la colaboración con otras plataformas como Slack para la integración de Firebase en las mismas, como se explica en "https://firebase.google.com/integrations/".
Hablando de los diferentes módulos que nos proporciona Firebase, los podemos dividir en 3 categorías.

- Desarrollo y compilación de apps. Dentro de esta categoría se han utilizado los siguientes módulos:
    - Cloud Firestore.
    - Cloud Functions.
    - Cloud Storage.

- Mejora de la calidad
- Herramientas de crecimiento. Dentro de esta categoría se han utilizado los siguientes módulos:
    - Google Analytics.
    - Cloud Messaging.

7.1.1. Módulo Cloud Firestore

Cloud Firestore es una base de datos flexible y escalable para la programación en servidores, dispositivos móviles y la Web desde Firebase y Google Cloud Platform. Mantiene tus datos sincronizados entre apps cliente a través de agentes de escucha en tiempo real y ofrece asistencia sin conexión para dispositivos móviles y la Web, por lo que puedes compilar apps con capacidad de respuesta que funcionan sin importar la latencia de la red ni la conectividad a Internet. Cloud Firestore también ofrece una integración sin interrupciones con otros productos de Firebase y Google Cloud Platform, incluido Cloud Functions.

Características clave:

- **Flexibilidad:** El modelo de datos de Cloud Firestore admite estructuras de datos flexibles y jerárquicas. Almacena tus datos en documentos, organizados en colecciones. Los documentos pueden contener objetos anidados complejos, además de subcolecciones.
- **Consultas expresivas:** En Cloud Firestore, puedes usar consultas para recuperar documentos individuales específicos o para recuperar todos los documentos de una colección que coinciden con los parámetros de la consulta. Tus consultas pueden incluir varios filtros en cadena y combinar los filtros con criterios de orden. También se indexan de forma predeterminada, por lo que el rendimiento de las consultas es proporcional al tamaño de tu conjunto de resultados, no del conjunto de datos.
- **Actualizaciones en tiempo real:** Cloud Firestore usa la sincronización de datos para actualizar los datos de cualquier dispositivo conectado. Sin embargo, también está diseñado para ejecutar consultas de recuperación únicas y sencillas de manera eficiente.
- **Asistencia sin conexión:** Cloud Firestore almacena en caché datos que usa tu app de forma activa, por lo que la app puede escribir, leer, escuchar y consultar datos, aunque el dispositivo se encuentre sin conexión. Cuando el dispositivo vuelve a estar en línea, Cloud Firestore sincroniza todos los cambios locales de vuelta a Cloud Firestore.
- **Diseñado para ajustarse a escala:** Cloud Firestore te ofrece lo mejor de la poderosa infraestructura de Google Cloud Platform: replicación automática de datos multiregión, garantías de coherencia sólida, operaciones atómicas por lotes y asistencia real sobre transacciones. Cloud Firestore está diseñada para controlar las cargas de trabajo de las bases de datos más complejas de las apps más grandes del mundo.

Toda esta información e información adicional necesaria se puede encontrar en el siguiente sitio web, "https://firebase.google.com/docs/firestore".

Este módulo lo empezó a utilizar el compañero que empezó la aplicación para desplegar la base de datos que utilizará la aplicación. Primeramente, se usó solo para guardar información relacionada con las plazas, pero debido al desarrollo de otros sistemas como la gestión de usuarios, se ha utilizado y profundizado más para un mejor uso entre el propio módulo y su conexión con otros.

7.1.2. Módulo Cloud Functions

Cloud Functions para Firebase es un framework sin servidores que te permite ejecutar de forma automática el código de backend en respuesta a las solicitudes HTTPS. Tu código JavaScript o TypeScript se almacena en la nube de Google y se ejecuta en un entorno administrado. No necesitas administrar ni escalar tus propios servidores.

Características clave:

- **Integra la plataforma de Firebase:** Las funciones que escribes pueden responder a eventos generados por varias funciones de Firebase y Google Cloud, desde activadores de Firebase Authentication hasta activadores de Cloud Storage. Integra en distintas funciones de Firebase con el SDK de Admin y Cloud Functions. Además, escribe tus propios webhooks para integrar con servicios de terceros. Cloud Functions minimiza el código estándar, lo que facilita el uso de Firebase y Google Cloud en tu función.
- **Sin mantenimiento:** Implementa tu código de JavaScript o TypeScript en los servidores con un comando desde la línea de comandos. Después de eso, Firebase aumenta los recursos de procesamiento automáticamente según los patrones de uso de los usuarios. No tendrás que preocuparte por las credenciales, la configuración de servidores, el aprovisionamiento de servidores nuevos ni por sacar de servicio los servidores antiguos.
- **Protege la privacidad y seguridad de tu lógica:** En muchos casos, los desarrolladores prefieren controlar la lógica de la aplicación en el servidor para evitar alteraciones del lado del cliente. A veces, no es recomendable permitir que se aplique ingeniería inversa a ese código. Cloud Functions está completamente aislada del cliente, de manera que puedes estar seguro de su privacidad y de que siempre hará exactamente lo que quieres.

Toda esta información e información adicional necesaria se puede encontrar en el siguiente sitio web, "https://firebase.google.com/docs/functions".

Este módulo se eligió por la necesidad de acceder a la base de datos de una forma externa, concretamente desde peticiones HTTP que son lanzadas desde las balizas de las plazas de aparcamiento.
Primeramente, se pensó en construir una API propia o usar la API que tiene el propio módulo de Cloud Firestore. Pero la primera opción requería demasiado trabajo y la segunda era una herramienta poco útil para nuestras necesidades. Por eso, decidimos utilizar este módulo, ya que con un sencillo despliegue tenemos acceso a todos los módulos de firebase utilizados.
Una vez que investigamos cómo funcionaba, nos resultó útil para construir peticiones HTTP, como he comentado anteriormente y, además, nos permitió implementar "listeners" en los documentos de los usuarios para recoger los cambios que se producen, y así poder conectarlo con el módulo de FCM y enviar notificaciones al móvil que esté usando el usuario.
Cabe mencionar que, debido a un cambio en la normativa de firebase, en marzo de 2021 este módulo dejará de dar soporte a las funciones escritas en node 8, siendo node 10 la versión a

la que se deberán de pasar. El problema de esto es que, la versión gratuita del proyecto de firebase, no permite crear funciones en node 10, eso es algo que requiere la licencia de pago, por lo que es muy posible que en pocos meses se deberá de solicitar dicha licencia para poder seguir utilizando dicho módulo.

7.1.3. Módulo de Cloud Storage

Cloud Storage para Firebase es un servicio de almacenamiento de objetos potente, simple y rentable construido para la escala de Google. Los SDK de Firebase para Cloud Storage agregan la seguridad de Google a las operaciones de carga y descarga de archivos para tus apps de Firebase, sin importar la calidad de la red. Puedes usar nuestros SDK para almacenar imágenes, audio, video y otros tipos de contenido generado por el usuario. En el servidor, puedes usar Google Cloud Storage para acceder a los mismos archivos.

Características clave:

- **Operaciones robustas:** Los SDK de Firebase para Cloud Storage realizan las operaciones de carga y descarga sin importar la calidad de la red. Las cargas y descargas son robustas, lo que significa que se reinician en el punto en el que se interrumpieron para así ahorrar tiempo y ancho de banda a los usuarios.
- **Seguridad sólida:** Los SDK de Firebase para Cloud Storage se integran con Firebase Authentication a fin de brindar autenticación intuitiva y sencilla para los programadores. Puedes usar nuestro modelo de seguridad declarativa para permitir el acceso según el nombre de archivo, el tamaño, el tipo de contenido y otros metadatos.
- **Gran escalabilidad:** Cloud Storage para Firebase está diseñado para escalar a exabytes si tu app se vuelve viral. Pasa fácilmente de la fase prototipo a la de producción con la misma infraestructura que respalda a Spotify y Google Fotos.

Toda esta información e información adicional necesaria se puede encontrar en el siguiente sitio web, "https://firebase.google.com/docs/storage".

Este módulo se eligió por la necesidad de tener un repositorio donde guardar las fotos de las tarjetas de movilidad reducida de los usuarios. Estuvimos investigando diferentes alternativas para su guardado como Dropbox u otra herramienta de Google como es Drive. Pero, debido a que estamos utilizando un gran número de módulos de Firebase, sería interesante utilizar el que nos proporciona para dicho fin, el llamado Cloud Storage.
Su funcionamiento es bastante similar al que nos ofrece otras herramientas, pero nos proporciona una potente y sencilla integración con otros servicios de Firebase, además, que tiene un SDK propio para su uso en Android, por lo que la programación se vuelve bastante sencilla.
La manera en la que implementamos este módulo consiste en crear una carpeta con el ID del usuario donde guardar todas las fotos que utilice en el registro o en la edición de perfil. Puede que parezca que estamos recopilando mucha información, pero en realidad estamos generando un rastro de un posible mal uso o usurpación de la información de otra persona. De esta manera, si se detecta un mal comportamiento del usuario, un administrador puede realizar las medidas oportunas como avisar al usuario afectado o borrar el perfil del usuario fraudulento.

7.1.4. Módulo de Google Analytics

Google Analytics es una solución de análisis ilimitada y gratuita que ocupa un lugar central en Firebase. Analytics se integra a distintas funciones de Firebase y te proporciona una capacidad ilimitada de generar informes sobre un total de hasta 500 eventos distintos que puedes definir con el SDK de Firebase. Los informes de Analytics te permiten entender claramente cómo se comportan tus usuarios para que puedas tomar decisiones fundamentadas en relación con el marketing de las apps y las optimizaciones del rendimiento.

Características clave:

- **Informes sin límites:** Analytics ofrece informes ilimitados de hasta 500 eventos distintos.
- **Segmentación del público:** Puedes definir públicos personalizados en Firebase console en función de los datos del dispositivo, eventos personalizados o propiedades del usuario. Estos públicos pueden usarse con otras funciones de Firebase para orientar funciones nuevas o mensajes de notificación.

Toda esta información e información adicional necesaria se puede encontrar en el siguiente sitio web, "https://firebase.google.com/docs/analytics".

Este módulo no se ha utilizado mucho durante el desarrollo, pero siempre es interesante tener integrada una herramienta que te permita ver el uso que se da a la aplicación, sobre todo para saber si al usuario final le gusta y la usa, además que permite identificar el tipo de usuario que la usa.

Toda esta información está disponible en la consola de Firebase, así que un administrador puede estar al día de toda esta información.

7.1.5. Módulo de Cloud Messaging

Firebase Cloud Messaging (FCM) es una solución de mensajería multiplataforma que te permite enviar mensajes de forma segura y gratuita.

Con FCM, puedes notificar a una app cliente que un correo electrónico nuevo o que otros datos están disponibles para la sincronización. Puedes enviar mensajes de notificación para volver a atraer a más usuarios y aumentar su retención. Para los casos prácticos de mensajería instantánea, un mensaje puede transferir una carga útil de hasta 4 KB a una app cliente.

Características clave:

- **Envía mensajes de notificación o mensajes de datos:** Envía mensajes de notificación que se muestran a tu usuario. También puedes enviar mensajes de datos y determinar completamente lo que ocurre en el código de tu aplicación.

- **Orientación versátil de mensajes:** Distribuye mensajes a tu app cliente en cualquiera de las siguientes tres formas: a dispositivos individuales, a grupos de dispositivos o a dispositivos suscritos a temas.
- **Envía mensajes desde apps cliente:** Envía mensajes de confirmación, de chat y de otros tipos desde los dispositivos a tu servidor a través del canal de conexión confiable de FCM que consume poca batería.

Toda esta información e información adicional necesaria se puede encontrar en el siguiente sitio web, "https://firebase.google.com/docs/cloud-messaging".

Este módulo se ha utilizado para el envío de notificaciones al usuario cuando se haya completado el proceso del registro automático de su ocupación en una plaza.
Para ello, hemos necesitado del uso del módulo de Cloud Functions, ya que como hemos comentado anteriormente, nos permite lanzar funciones cuando ocurre una actualización en el documento de un usuario de la base de datos. Cuando este evento ocurre, lanzamos una notificación al usuario a través del token que representa su móvil guardado en la base de datos.

7.2. Comunicación con las balizas

En este apartado se detalla todo el proceso que se lleva a cabo para la comunicación de nuestra aplicación móvil con las balizas que se encuentran en las diferentes plazas de aparcamiento.

Cabe mencionar que este aspecto ha sido uno de los más complicados a la hora de pensar cómo hacerlo de la mejor manera posible.
Al principio estuvimos viendo si era mejor que el móvil escaneara beacons que emitiera la baliza o viceversa, finalmente decidimos que lo mejor era que el móvil transmitiera porque la baliza que disponemos emite señal bluetooth general, no está codificada como beacons. Además, hacer que el móvil estuviera escaneando supondría un mayor gasto energético que la transmisión, a parte del proceso del tratamiento de datos recibido y la comunicación con el servidor.
Una vez decidido el diseño para la funcionalidad a la que van dedicadas estas herramientas, el registro automático de la ocupación de una plaza por un usuario conocido, compuse el diagrama de secuencia que ilustra este proceso, el cual está expuesto y explicado en el apartado de diseño de este documento. Siguiendo este diagrama tenía que pensar cómo podía distribuir y comprimir el ID del usuario para que el beacon fuera capaz de transmitirlo.

Después de muchas pruebas fallidas por falta de espacio o por la necesidad de formato de cada campo que compone el Beacon encontré una forma, un tanto engorrosa pero efectiva, para poder transmitir el ID del usuario.

Antes de explicar este proceso, hay que comentar qué son y qué estructura tienen los Beacons, concretamente la del AltBeacon que es el que voy a utilizar.

AltBeacon es una especificación de protocolo que define un formato de mensaje para anuncios de balizas de proximidad. Los anuncios de balizas de proximidad AltBeacon son transmitidos por dispositivos con el fin de señalar su proximidad a receptores cercanos. El contenido del mensaje emitido contiene información que el dispositivo receptor puede utilizar para identificar la baliza y calcular su distancia relativa a la baliza. El dispositivo receptor puede usar esta información como un disparador contextual para ejecutar procedimientos e implementar comportamientos que sean relevantes para estar cerca de la baliza transmisora.

Los ejemplos de casos de uso de balizas de proximidad incluyen, entre otros:

- Notificar a los usuarios sobre ofertas especiales cuando visitan áreas dentro de una tienda departamental.
- Presentar oportunidades para explorar información adicional sobre una exhibición a un visitante del museo.
- Registro automático con el sistema de reservas de un restaurante cuando llega el cliente.

7.2.1. Requisitos de implementación

La funcionalidad de la baliza de proximidad AltBeacon no se limita a los dispositivos de función única, sino que puede incorporarse como una característica de cualquier dispositivo que sea compatible con Bluetooth Low Energy y que cumpla con los requisitos definidos en la Especificación de Bluetooth Versión 4.0, Volumen 0, Parte B, Sección 4.4 Configuración de núcleo de baja energía o Sección 4.5 Configuración de núcleo combinado de tarifa básica y baja energía.

Los anuncios de AltBeacon se encapsulan como la carga útil de una publicidad no dirigida no conectable PDU (ADV_NONCONN_IND) como se define en la Especificación de Bluetooth Versión 4.0, Volumen 6, Parte B, Sección 2.3 PDU del canal de publicidad.

Los dispositivos que transmiten paquetes de anuncios de balizas de proximidad se denominan anunciantes. Los dispositivos que reciben anuncios de balizas de proximidad se denominan escáneres. Estos roles siguen las convenciones definidas en la Especificación de Bluetooth Versión 4.0, Volumen 1, Parte A, Sección 1.2 Descripción general del funcionamiento de Bluetooth de baja energía.

7.2.2. Formato del protocolo AltBeacon

El anuncio de AltBeacon hace uso de la estructura de datos publicitarios específicos del fabricante según se define en la especificación Bluetooth versión 4.0, volumen 6, parte B, sección 2.3 PDU del canal publicitario.

El anuncio de AltBeacon se compone de un campo de 1 byte de longitud, un campo de tipo de 1 byte y un identificador de empresa de dos bytes, según lo prescrito por el formato de estructura de datos publicitarios específicos del fabricante, seguido de 24 bytes adicionales que contienen los datos del anuncio de la baliza.

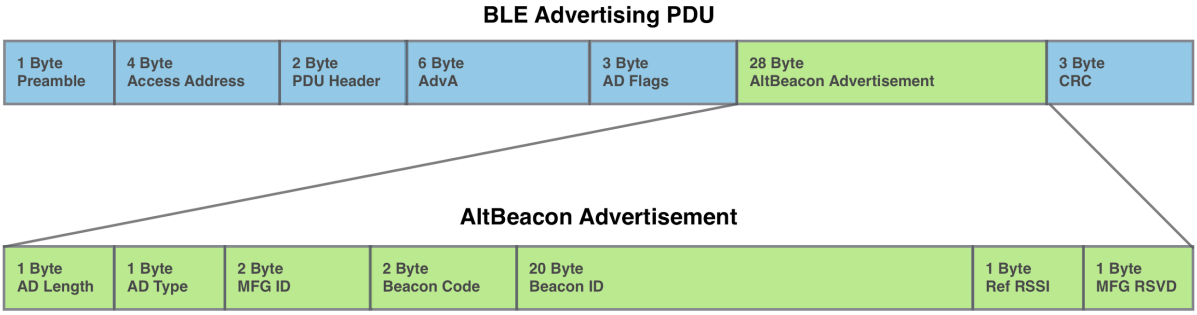


Figura 20 - Estructura del AltBeacon Advertisement

7.2.3. Diagrama del protocolo

Offsets	Octet	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	AD LENGTH								AD TYPE								MFG ID															
4	32	BEACON CODE																BEACON ID															
8	64	BEACON ID (CONTINUED)																															
12	96	BEACON ID (CONTINUED)																															
16	128	BEACON ID (CONTINUED)																															
20	160	BEACON ID (CONTINUED)																															
24	192	BEACON ID (CONTINUED)																REFERENCE RSSI								MFG RESERVED							

Figura 21 - Diagrama del protocolo AltBeacon

7.2.4. Campos del protocolo

Nombre del campo	Descripción	Valores aceptados
LONGITUD DEL ANUNCIO [MFG ESPECÍFICO]	Longitud del tipo y la parte de datos de la estructura de datos publicitarios específicos del fabricante.	`0x1B`
TIPO DE ANUNCIO [MFG ESPECÍFICO]	Tipo que representa la estructura de datos publicitarios específicos del fabricante.	`0xFF`
ID de MFG	El código de identificación de la empresa del fabricante del dispositivo de baliza.	La representación little endian del código de empresa del fabricante del dispositivo de baliza según lo mantiene la base de datos de números asignados de Bluetooth SIG
CÓDIGO DE BALIZA	El código de publicidad AltBeacon	La representación big endian del valor `0xBEAC`
ID DE BALIZA	Un valor de 20 bytes que identifica de forma única la baliza	La representación big endian del identificador de baliza. Para fines de interoperabilidad, los primeros 16 bytes o más del identificador de baliza deben ser únicos para la unidad organizativa del anunciante. Los bytes restantes del identificador de baliza se pueden subdividir según sea necesario para el caso de uso.

REFERENCIA RSSI	Un valor de 1 byte que representa la intensidad media de la señal recibida a 1 m del anunciante.	Un valor de 1 byte con signo de 0 a -127
MFG RESERVADO	Reservado para su uso por el fabricante para implementar características especiales	Un valor de 1 byte desde `0x00`hasta `0xFF`. La interpretación de este valor debe ser definida por el fabricante y debe evaluarse en función del `MFG IDvalor`

Una vez explicado todo lo relacionado con los AltBeacons, voy a detallar cómo he usado la estructura de los beacons para la implementación de los mismos en el proyecto.

Para la emisión de Beacons desde el dispositivo Android he utilizado la librería “android-beacon-library”, la cual explicaré más adelante en el apartado de Tecnologías utilizadas. Con esta librería podemos elegir qué tipo de Beacon queremos transmitir. Podemos elegir entre los siguientes formatos:

- ALTBEACON: "m:2-3=beac, i:4-19, i:20-21, i:22-23, p:24-24, d:25-25".
- EDDYSTONE TLM: "x, s:0-1=feaa, m:2-2=20, d:3-3, d:4-5, d:6-7, d:8-11, d:12-15".
- EDDYSTONE UID: "s:0-1=feaa, m:2-2=00, p:3-3:-41, i:4-13, i:14-19".
- EDDYSTONE URL: "s:0-1=feaa, m:2-2=10, p:3-3:-41, i:4-20v".
- IBEACON: "m:2-3=0215, i:4-19, I:20-21, I:22-23, p:24-24".

Dichos formatos pertenecen a protocolos Open Source (AltBeacon), de Google (Eddystone) y de Apple (iBeacon), los cuales están seguidos por el formato teniendo su estructura en bytes, las cuales debemos indicar a la hora de transmitir para que la librería sepa qué tipo de Beacon tiene que utilizar.

Como ya he comentado en diferentes ocasiones, el formato utilizado para nuestro proyecto es el AltBeacon, el cual dispone en su apartado de identificación 3 subcategorías.

Estas categorías son:

- UUID: cadena de 32 caracteres en hexadecimal separada por guiones de la siguiente manera: “FFFFFFFF-FFFF-FFFF-FFFF-FFFFFFFFFFFF”. Esta categoría es la referenciada por la primera i indicada en su formato, dicha categoría consta de 16 bytes.
- Mayor: cadena de hasta 65535 caracteres en formato decimal (solo incluyendo enteros), ya que es el número máximo que admiten los dos bytes que ocupa.
- Minor: es otra categoría, pero sigue la misma estructura que el mayor.

Para entender mejor la implementación y el uso que se ha dado a estas subcategorías, voy a detallar un ejemplo en donde uso un ID de un usuario para ver las partes y las conversiones que se deben hacer para poder enviar y recibir dicha información en el AltBeacon.

7.2.5. Ejemplo de implementación de AltBeacons

Prueba transmisión del userID a través del Beacon:

- UserID en firebase: **8Njg7LhQIlcTRLW9b45R**
- UserID emitido por el beacon:
    - UUID: 384e6a67-374c-6851-496c-6354524c5739 (corresponde en **hexadecimal** a *8Njg7LhQIlcTRLW9* del userID real, ya que son los 32 bytes que admite este campo).
    - Major: 25140 (corresponde en **decimal** a *b4* del userID real, ya que este campo solo admite 2 bytes).
    - Minor: 13650 (corresponde en **decimal** a *5R* del userID real, ya que este campo solo admite 2 bytes).

- Transformamos el major a hexadecimal → 25140 (**decimal**) = 6234 (**hexadecimal**).
- Transformamos el minor a hexadecimal → 13650 (**decimal**) = 3552 (**hexadecimal**).

Una vez escaneado el Beacon, la baliza deberá:

- Construimos el userID entero en **hexadecimal** uniendo cada parámetro y siguiendo el siguiente orden: UUID (sin guiones) + major + minor: 384e6a67374c6851496c6354524c5739 + 6234 + 3552 = 384e6a67374c6851496c6354524c573962343552
- Transformamos la cadena de caracteres en **hexadecimal** en cadena de **texto** para obtener el userID en el mismo formato en el que lo tenemos guardados en la base de datos: 384e6a67374c6851496c6354524c573962343552 → **8Njg7LhQIlcTRLW9b45R**

7.3. Componentes implementados

En este apartado se detallarán los componentes de software que se han utilizado para la implementación del sistema, incluyendo los de Android, lógica del sistema, lógica de los datos y conexiones con los servicios.

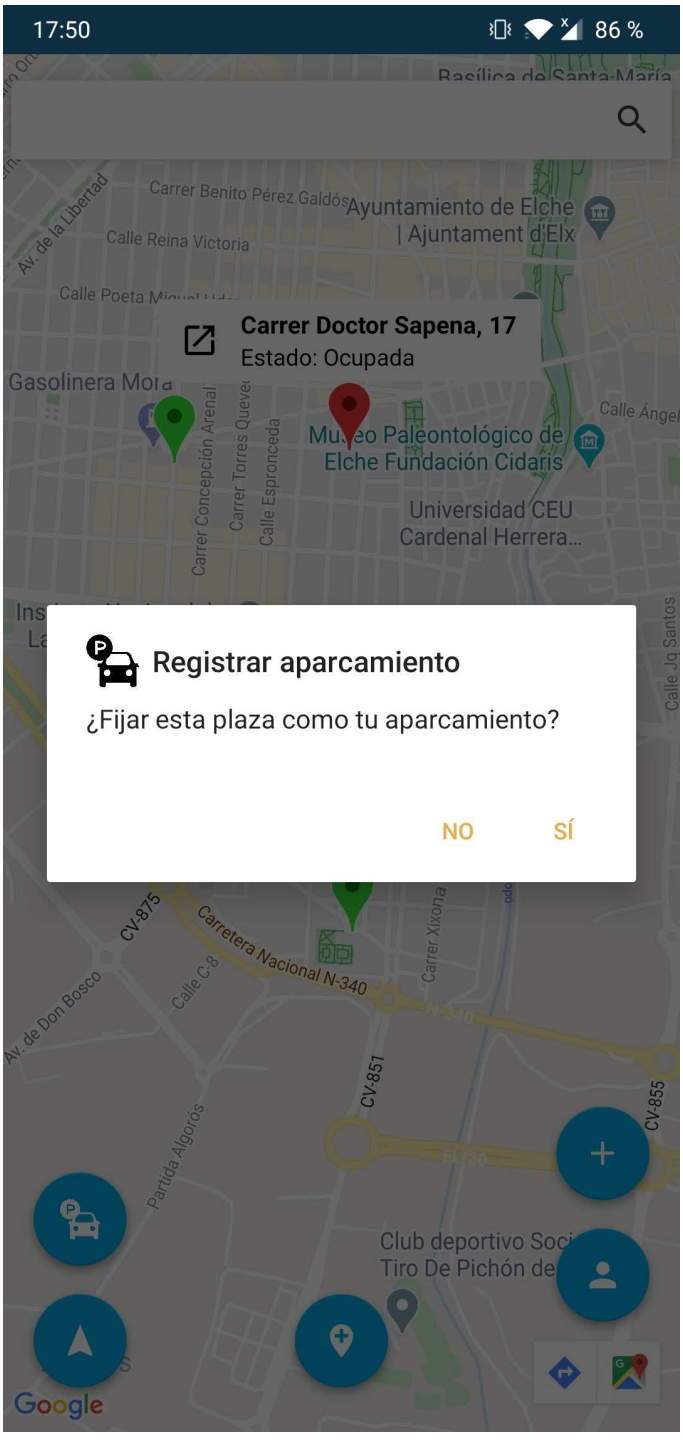
En la siguiente tabla se puede ver en modo resumen los subapartados que se van a explicar junto con una breve descripción para tener una idea sobre qué tratan:

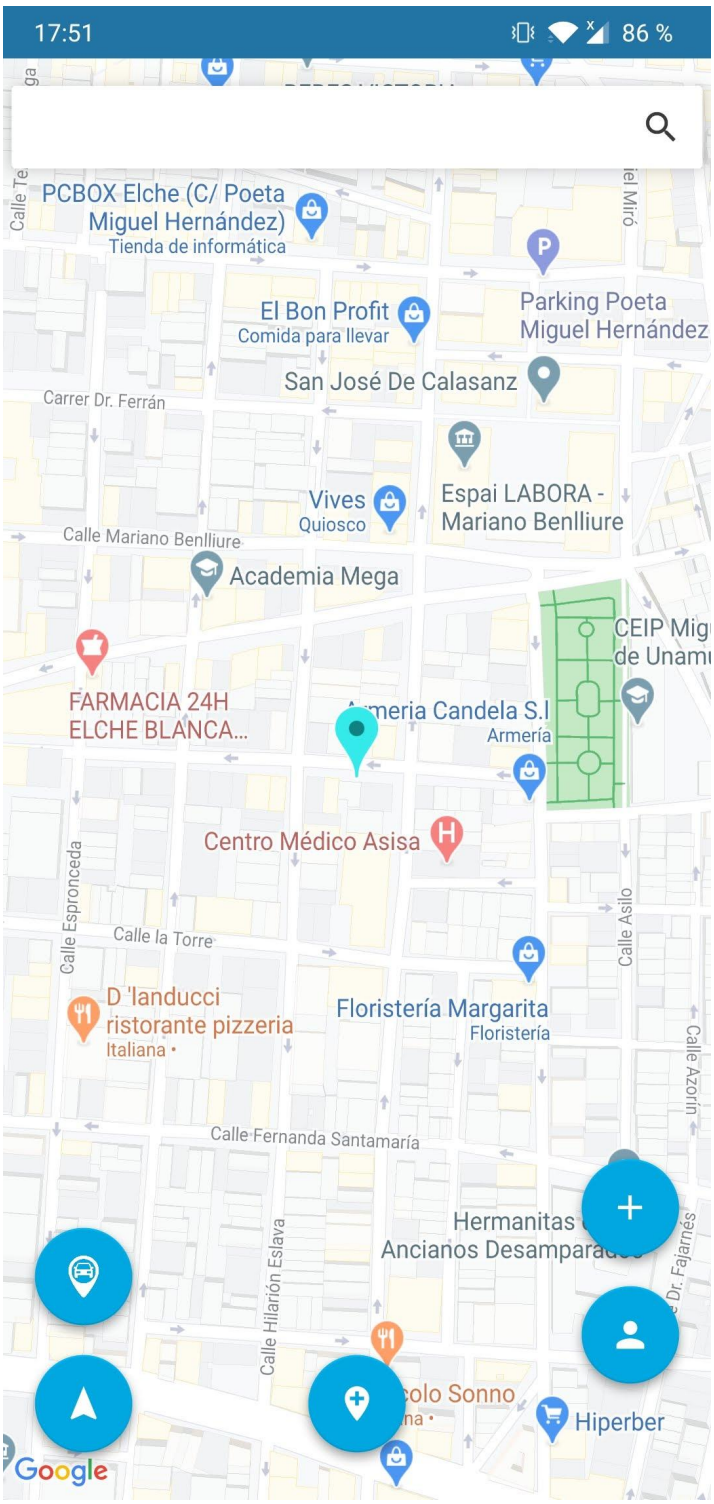
Actividades	Representan las diferentes pantallas que dispone la aplicación
Clases del modelo de datos	Representa las diferentes clases que posee la aplicación como representación de los datos de la base de datos junto con una serie de métodos para gestionarlos
Receiver	Representa la clase que nos permite recibir datos de manera externa y abrir la aplicación para poder procesarlos
Servicio de notificaciones	Representa a las clases que nos permiten recibir y gestionar los datos del servidor relacionados con las notificaciones que muestra la aplicación al usuario
Funciones en la nube	Representa a las distintas funciones encargadas de la comunicación de la baliza con el servidor y realizar las operaciones pertinentes en el servidor.

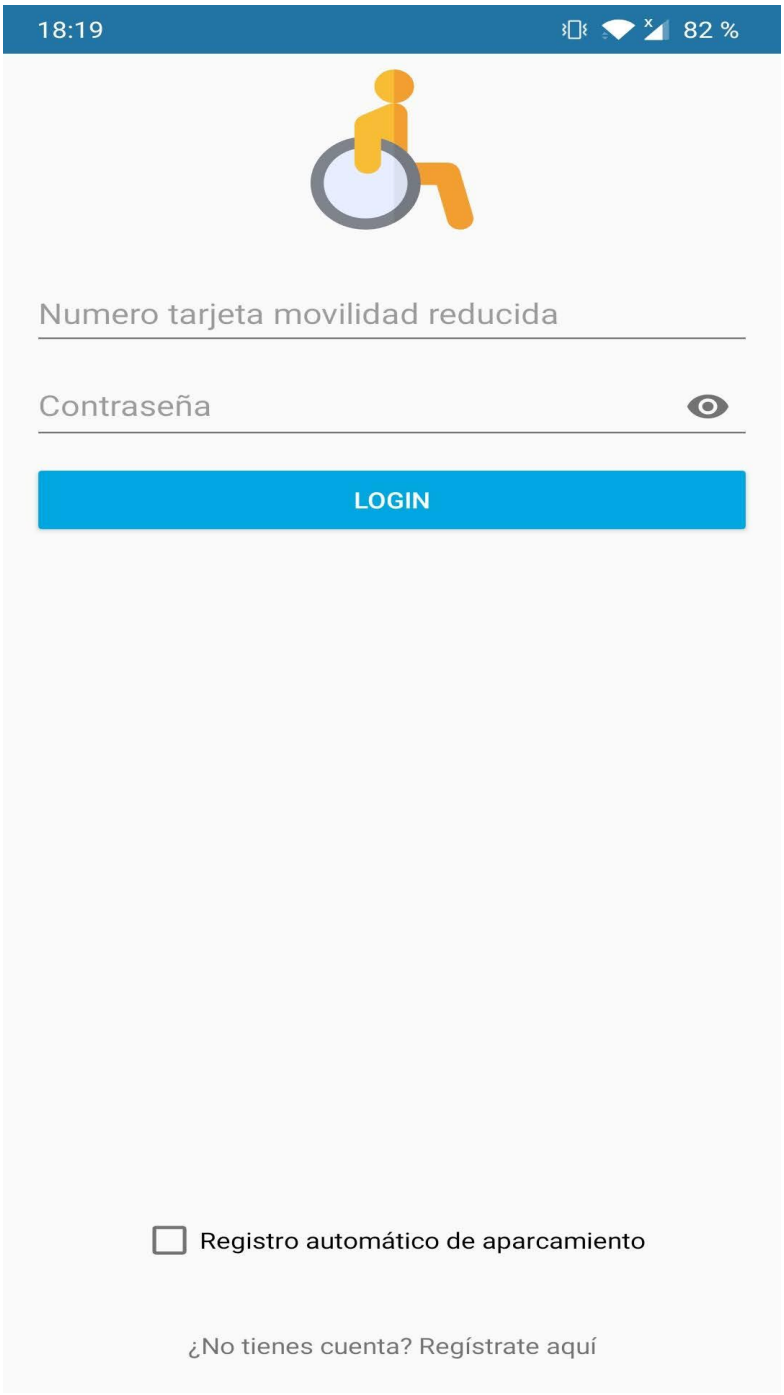
7.3.1. Actividades

En esta sección voy a explicar las diferentes actividades que al final se han implementado. Para la construcción de estas pantallas he tenido en cuenta todo lo que está descrito en el apartado de diseño, concretamente en el apartado de los bocetos, ya que me han ayudado a consolidar la idea que allí se detalla.

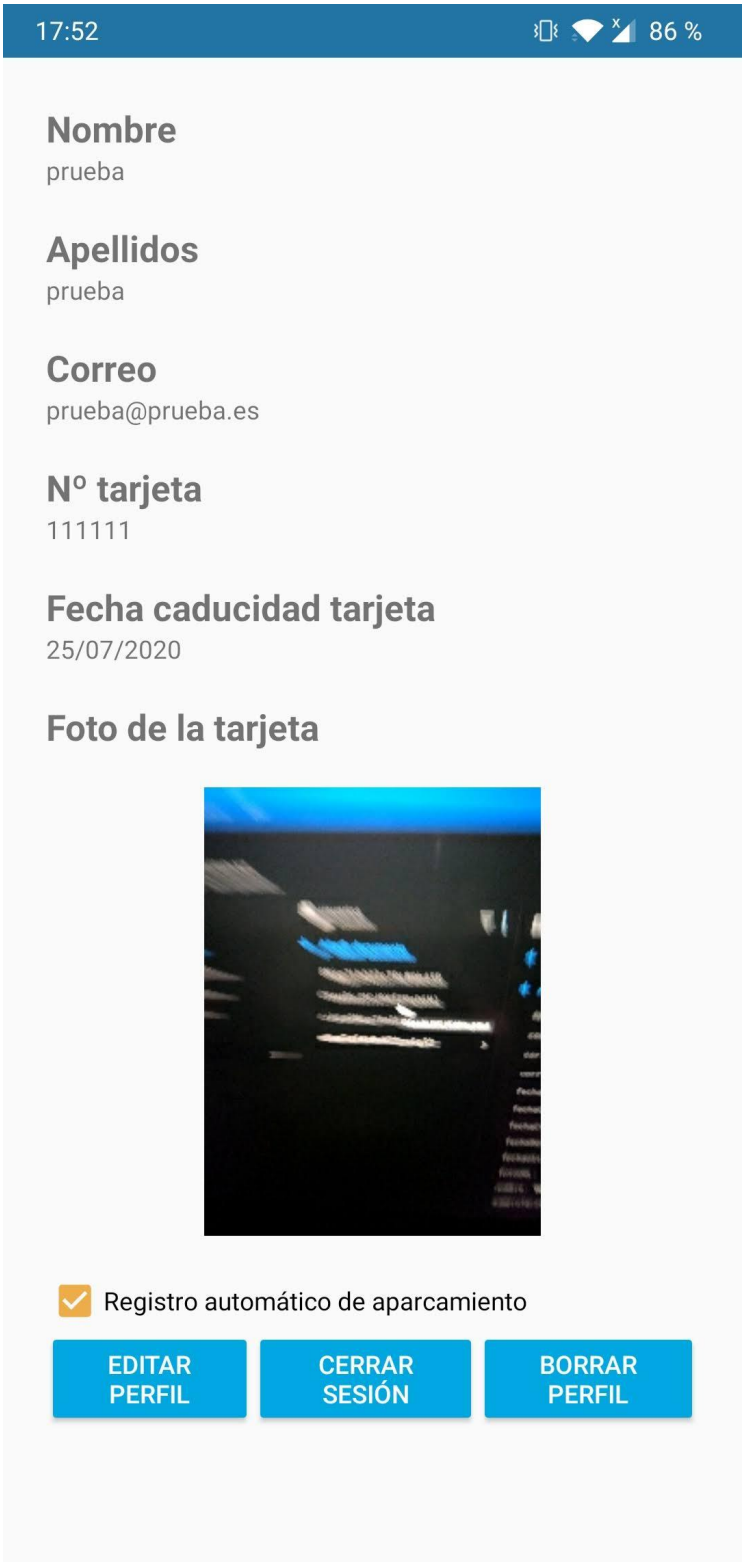
Para no repetir información, me ceñiré únicamente a aquellas actividades que haya implementado yo, dejando de lado las que desarrolló el compañero que empezó el proyecto. Es decir, trataré las actividades relacionadas con la gestión de usuarios y comentaré las novedades implementadas en la gestión de plazas.

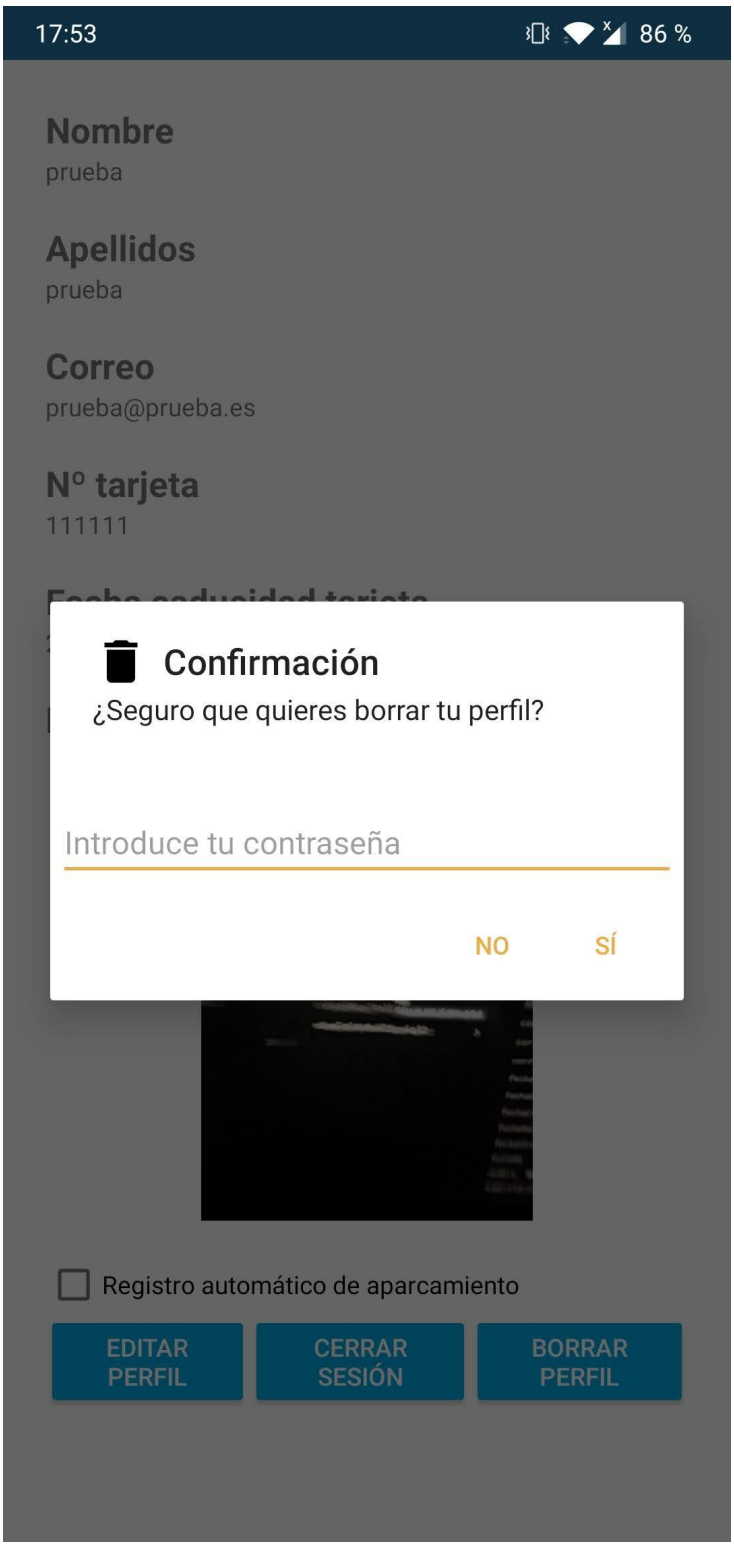
Nombre	MapsActivity
Descripción	Actividad donde podremos acceder a la pantalla de login o perfil, en función de si tenemos una sesión iniciada o no. También podremos registrar nuestra ocupación de forma manual al tocar una plaza ocupada, ya que aparece un nuevo botón abajo a la izquierda que nos permite hacerlo. En el mismo lugar puede aparecer otro botón que nos lleve a nuestra plaza ocupada si está registrada a nuestro nombre.
Resultado	


17:51
86 %
PCBOX Elche (C/ Poeta Miguel Hernández)
Tienda de informática
El Bon Profit
Comida para llevar
Parking Poeta Miguel Hernández
San José De Calasanz
Carrer Dr. Ferrán
Vives
Quiosco
Espai LABORA - Mariano Benlliure
Calle Mariano Benlliure
Academia Mega
FARMACIA 24H ELCHE BLANCA...
Armería Candela S.l
Armería
Centro Médico Asisa
Calle Espronceda
Calle la Torre
Calle Asilo
D 'Ianducci ristorante pizzeria
Italiana
Floristería Margarita
Floristería
Calle Azorín
Calle Fernanda Santamaría
Calle Hilarión Eslava
Hiperber
Google

Nombre	LoginActivity
Descripción	Actividad donde podremos realizar el proceso de login introduciendo nuestro número de tarjeta de movilidad reducida y nuestra contraseña. Si los datos son erróneos, el usuario será notificado. Si el proceso de login se realiza satisfactoriamente, el perfil del usuario en la base de datos se actualiza con el token del dispositivo donde haya iniciado sesión por última vez.
Resultado	

Nombre	SingUpActivity
Descripción	Actividad donde podremos realizar el proceso de registro. Para ello debemos rellenar todos los datos del formulario. Si falta algún dato por introducir será notificado al usuario. Si el número de tarjeta está ya registrado será notificado al usuario. El proceso de registro consiste en crear el nodo del nuevo usuario en la base de datos, subir la foto de su tarjeta al módulo de almacenamiento asociado al id generado al crear dicho usuario en la base de datos, se actualiza el perfil del usuario añadiendo la URL para acceder a la foto y se añade el token del dispositivo donde se esté registrando el dispositivo.
Resultado	

Nombre	UserProfileActivity
Descripción	Actividad donde podremos ver los datos del perfil del usuario, cerrar sesión, acceder a la pantalla de edición del perfil y borrado del perfil. Para poder borrar su perfil, el usuario deberá volver a introducir su contraseña para validar dicho proceso.
Resultado	


17:53
86 %
Nombre
prueba
Apellidos
prueba
Correo
prueba@prueba.es
Nº tarjeta
111111
Confirmación
¿Seguro que quieres borrar tu perfil?
Introduce tu contraseña
NO
SÍ
Registro automático de aparcamiento
EDITAR PERFIL
CERRAR SESIÓN
BORRAR PERFIL

Nombre	UserEditActivity
Descripción	Actividad donde podremos realizar el proceso de edición del perfil. Los datos que se pueden modificar dependen del estado de verificación del perfil y de la caducidad de los propios datos, como he comentado anteriormente.
Resultado	

7.3.2. Clases del modelo de datos

En este apartado voy a describir las clases que se utilizan para manejar los datos obtenidos de la base de datos en la aplicación. Estas clases son "PlazaParking" y "Usuario".
La clase "PlazaParkingDB" fue creada por el compañero que empezó el proyecto para manejar los datos que se guardan de las plazas de aparcamiento, por ese motivo, dicha clase contiene como atributos dichos datos y métodos (getters, setters y constructor) para el manejo de los mismos.

Por otra parte, he creado la clase "Usuarios" que tiene el mismo objetivo que la clase anterior. La diferencia es que, en este caso, está destinada a manejar los datos de los usuarios en la aplicación.

Cabe mencionar que las dos clases descritas en este apartado solo nos ayudan a encapsular los datos. Para poder recibir dichos datos de Firestore, Firebase nos proporciona un API con el que poder realizar operaciones con dicha base de datos. Todas estas operaciones parten de la clase "FirebaseFirestore", que con el método "*getInstance()*" obtenemos la referencia a la base de datos asociada a nuestro proyecto, y desde ahí ya tenemos métodos para consulta, actualización, borrado, entre otros.

En el caso de las consultas, disponemos del método "*collection ("nombre de la colección a la que queramos acceder")*" con el que obtenemos una referencia a la colección mencionada por parámetros y desde ahí ya podemos acceder a toda ella o, a través de otros métodos en modo de filtro, podemos acceder a solo unos cuantos documentos. Al igual que el resto de las operaciones, las consultas disponen de una serie de métodos que hacen el papel de "escuchadores", los cuales se llaman cuando se recibe una respuesta del servidor. Tenemos varios métodos de ese tipo para poder controlar simplemente la respuesta, o si queremos saber si la petición ha sido satisfactoria o si ha dado algún error, y así poder controlarlo. En el caso de recibir éxito en la petición, la misma respuesta nos devuelve los datos solicitados, pero sin procesar, ya que sigue un formato estándar para generalizar el procedimiento. Es en este caso, donde son de ayuda las clases que he descrito en el inicio de este apartado, ya que nos permiten dar sentido en el código a los datos recibidos, además de darnos un mejor control sobre ellos.

7.3.3. Receiver

Para poder acceder a la aplicación estando en background he necesitado construir un Broadcast receiver llamado "TransitionsReceiver".

Para poder usar esta clase es necesario declararla en el manifiesto, al igual que ocurre con las actividades o servicios, por ejemplo.

Una vez todo declarado, ya podemos usarla. Su función principal es recibir las respuestas que nos manda el API de reconocimiento de la actividad que tenemos declarado y configurado para detectar el cese de conducir.

Cuando dicho API detecta la transición de actividad correspondiente con dejar de conducir, se llama al Receiver para que compruebe la actividad recibida y empiece a emitir beacons. Para hacer pruebas, a parte de la transición de dejar de conducir, he implementado otras transiciones de actividad como son empezar o dejar de andar, para que resultase más sencillo probar esta funcionalidad.

Cabe comentar, que esta funcionalidad se podría implementar creando un servicio que fuera llamado desde el receiver, ya que el sistema operativo podría cortar el proceso. Aunque en nuestro caso, esto no era necesario, ya que el receiver simplemente tendría que empezar la emisión de los beacons, y la librería que lo gestiona nos da un margen de en torno a 10 - 15 min de tiempo de emisión. Este punto es importante, por si en un futuro, fuera necesario manejar con más detalle el proceso de emisión, y de esta manera se podría solucionar.

7.3.4. Servicio de notificaciones

Para implementar el servicio de notificaciones he añadido al proyecto el módulo de notificaciones de Firebase FCM. Para poder utilizar este servicio es necesario declararlo en el manifiesto.

Una vez declarado todo, ya podemos recibir notificaciones procedentes de Firebase. En nuestro caso, no será directamente la consola de Firebase quien envíe las notificaciones, sino que será una función de Cloud Functions, quien se encargará de ello. Debido a esto, es necesario utilizar un sistema de "tokens" para indicar a qué dispositivo tiene que enviar la notificación, por lo que este dato se guarda en el proceso de registro y se actualiza en el login.

Cabe mencionar que el token es una cadena de texto que identifica la aplicación en el dispositivo, por lo que cada instalación generará un token nuevo. Además, si se desinstala la aplicación y se vuelve a instalar, o simplemente se borran los datos de la aplicación, se generará un token nuevo.

A parte de la recepción básica de notificaciones, es decir, notificaciones solo con título y sin cuerpo para que sea la bandeja de notificaciones del sistema operativo quien lo gestione, se ha implementado la clase "MyFirebaseMessagingService" por si en un futuro se requiere controlar la recepción de notificaciones con cuerpo o con la aplicación en primer plano, y así, recoger los datos de la notificación y realizar las operaciones pertinentes.

7.3.5. Funciones en la nube

Para la comunicación entre la baliza y el servidor he implementado unas funciones utilizando el módulo de Cloud Functions de Firebase. He construido 3 funciones, en JavaScript, llamadas "registrarOcupacionPlaza", "registrarLiberacionPlaza" y "enviarNotificacionAparcamiento".

- **registrarOcupacionPlaza**: Esta función HTTP espera recibir la ID de la plaza ocupada y el ID del usuario que la ha ocupado. El proceso que sigue dicha función depende de si recibe el ID de la plaza y el del usuario, o solo el de la plaza. Si recibe ambos ID comprueba que existan, por si hubiera algún error en el envío, y actualiza el estado de dicha plaza, usuario y, si tuviera una plaza anterior registrada el usuario, la libera. Si solo recibe el ID de la plaza, comprueba que dicha plaza exista, y si existe, marca su ocupación.
- **registrarLiberacionPlaza:** Esta función HTTP recibe el ID de la plaza que se ha liberado. Si dicha plaza existe, actualiza su estado de ocupación como libre y actualiza el perfil del usuario que la está ocupando, si se sabe quién es, indicando que ya no está ocupando ninguna plaza.
- **enviarNotificacionAparcamiento:** Esta función sirve para enviar una notificación al usuario cuando se registra su ocupación. Se activa cada vez que ocurre algún cambio en algún documento del nodo de usuarios. Se comprueban los datos de antes y después de la modificación que ha provocado la llamada de esta función. Solo si la plaza anterior registrada es diferente a la nueva y la nueva no es nula, se procede a la construcción y envío de la notificación indicando en el título de la notificación que se ha registrado la plaza correctamente.

Estas 3 funciones no forman parte de la aplicación móvil en sí, sino que hacen las funciones de una especie de API REST alojado en el servidor de Firebase, el cual la baliza puede utilizar para actualizar la base de datos. Debido a esto, las funciones de "registrarOcupacionPlaza" y "registrarLiberacionPlaza" están diseñadas para ser llamadas de forma externa a través de una petición HTTP, la cual tiene un cuerpo donde se adjuntan los IDs.
A diferencia de las 2 anteriores, la función "enviarNotificacionAparcamiento" no se llama desde una petición HTTP. Está diseñada para que sea el mismo servidor la que la llame cuando ocurra algún cambio en los datos de la plaza.

7.4. Tecnologías utilizadas

En este apartado se detallarán todas aquellas tecnologías que son necesarias para implementar las funcionalidades descritas en este documento:

- **Java**: Esta aplicación está destinada a dispositivos móviles con sistema operativo Android, por lo que está desarrollada en Java con algunas variaciones en el lenguaje y estructuración de datos orientados a este sistema operativos, como son los activities, manejo de layouts, entre otras. Es cierto que, actualmente, se pueden desarrollar aplicaciones nativas en Android utilizando un lenguaje creado específicamente para ello llamado Kotlin, el cual te permite hacer lo mismo que puedes hacer con Java pero con menos líneas de código. Pero debido a que el proyecto ya estaba creado en Java y que en el máster desarrollamos en dicho lenguaje, seguí usándolo. Todo esto no impide que, en un futuro, la aplicación pueda pasarse a Kotlin.

- **XML:** Para el diseño de las interfaces que mostrarán las diferentes pantallas se utiliza el lenguaje basado en etiquetas XML. No ha habido una motivación adicional para elegirlo como lenguaje de estructuración de componentes, más que es el lenguaje que se utiliza por defecto para realizar esta tarea en Android Studio. A pesar de esto, este lenguaje se integra muy bien con esta herramienta ya que está preparada para ello, hasta dispone de un simulador gráfico para ir viendo en tiempo real aquellos layouts que vamos diseñando. La manera de trabajar con XML es muy similar a otros lenguajes basados en etiquetas, solo necesitamos estructurar diferentes componentes con una serie de atributos que harán de sus propiedades.

- **JavaScript, JS.** Es un lenguaje de programación interpretado. Es utilizado en el lado cliente aumentando la experiencia de los usuarios con páginas web. Aunque en nuestro caso, lo utilizaremos para construir las diferentes funciones destinadas a Firebase Cloud Functions con ayuda de NodeJS, concretamente y de momento en NodeJS 8.

- **SDK Firebase:** En este punto agrupo todas aquellas dependencias que necesitamos para implementar todos los módulos de Firebase que necesitemos. Este SDK integra todas aquellas clases que permiten desarrollar en Java las funcionalidades anteriormente descritas.

- **Google Maps:** En este apartado agrupo las diferentes librerías de Google Maps que hemos integrado en la aplicación para el uso de visualización y gestión de mapas, servicios de localización, búsqueda de direcciones, entre otros.

- **Android Beacon Library:** Esta es la librería que he utilizado para la inclusión de toda la funcionalidad relacionada con los Beacons. Nos permite escanear y transmitir Beacons, aunque en nuestro caso, solo nos hacía falta la parte de transmitir Beacons. Es bastante útil, ya que dispone de numerosos métodos y clases para poder configurar diferentes tipos de Beacons que emitir, configurar su emisión, su estructura y los datos que lleva consigo. En nuestro caso, emitimos Beacons del tipo AltBeacons, ya que dispone de mayor capacidad (bytes) para poder cargar datos, siendo estos el ID del usuario fraccionado y transformado en sus diferentes campos, como he explicado anteriormente.

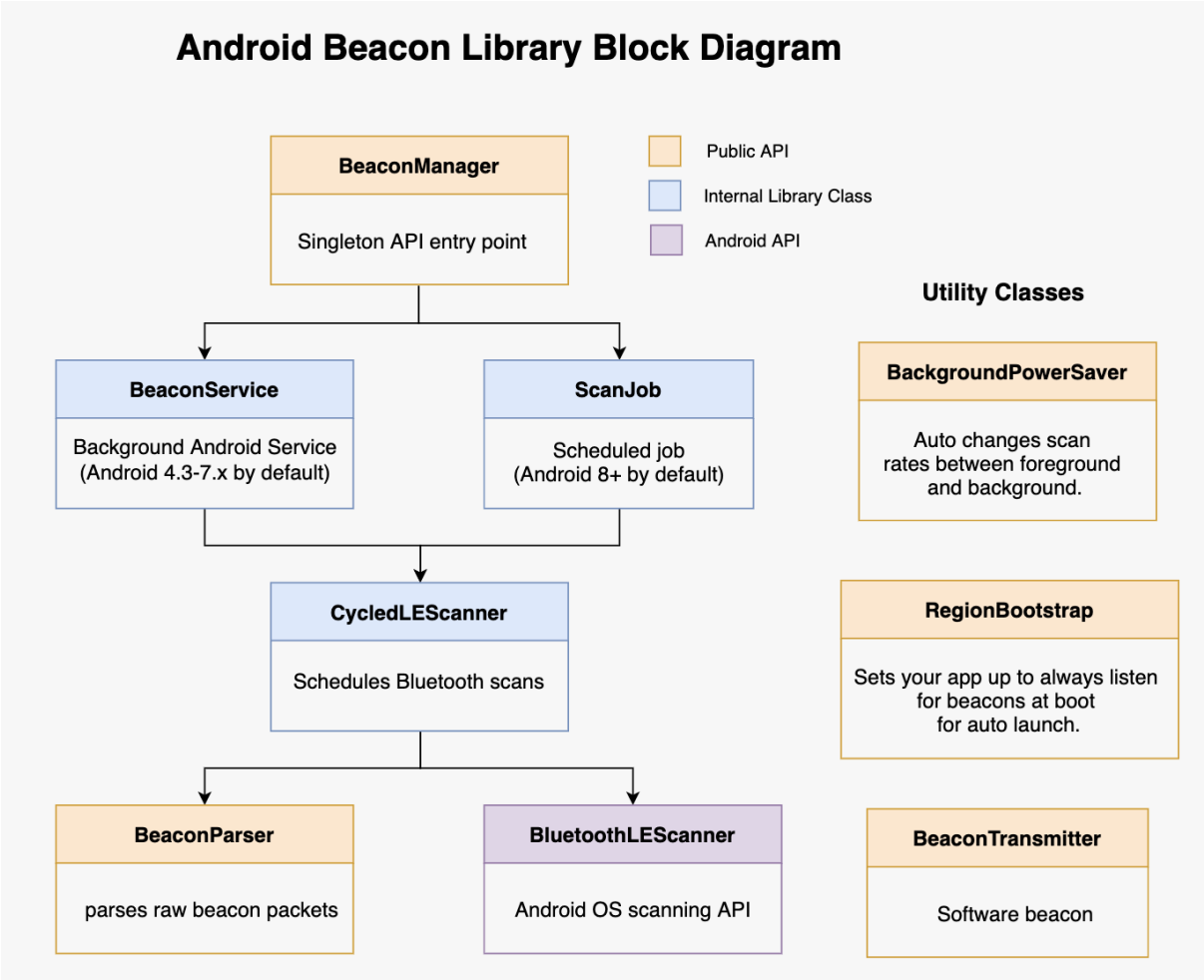


Figura 22 - Estructura de Android Beacon Library

- **Reconocimiento de actividad.** Esta herramienta, desarrollada por Google para Android, nos proporciona una cómoda forma de reconocer, a través de varios sensores del dispositivo, el tipo de actividad que está teniendo el usuario con su móvil. A través de un proceso interno, es capaz de dar una probabilidad de cuál es la actividad más probable que esté realizando el usuario de entre un grupo de ellas, el cual está formado por: en bici, conduciendo, de pie, inclinado, corriendo, andando, parado o desconocido.
  Para nuestro uso, más que reconocer la actividad más probable, lo que nos interesa es recoger cambios en las actividades, concretamente el de dejar de conducir, ya que es el momento en el que empieza toda la funcionalidad (emisión) de los Beacons.

7.5. Herramientas utilizadas

En este apartado se definen las herramientas que se utilizan para el desarrollo del sistema:

- **Android OS:** Sistema operativo basado en Linux. Fue diseñado principalmente para dispositivos móviles con pantalla táctil (Smartphones, tablets o relojes). Es un sistema operativo muy intuitivo, lo que le ha llevado a tener la mayor cuota de mercado.

- **Android Studio** (versión 3.6.3): Es un entorno de desarrollo integrado oficial para la plataforma Android. Proporciona las herramientas para crear aplicaciones en cada tipo de dispositivo Android. Posee soporte para desarrolladores y gran cantidad de información (guías y documentación).

- **Firebase Console:** Es un entorno de administración de proyectos Firebase y de todos los módulos que integre. Con una interfaz muy sencilla nos proporciona información sobre la configuración, implementación y uso de dichos proyectos. En nuestro caso, lo he utilizado para depurar y controlar las diferentes funciones que iba probando. Como, por ejemplo, las funciones en Android Studio para la carga y descarga de datos entre servidor y la aplicación, como para ver el funcionamiento de las Cloud Functions.

- **Visual Studio Code:** Es un editor de código fuente desarrollado por Microsoft para Windows, Linux y macOS. Incluye soporte para la depuración, control integrado de Git, resaltado de sintaxis, finalización inteligente de código, fragmentos y refactorización de código. También es personalizable, por lo que los usuarios pueden cambiar el tema del editor, los atajos de teclado y las preferencias. Es gratuito y de código abierto, aunque la descarga oficial está bajo software privativo e incluye características personalizadas por Microsoft. En nuestro caso, lo he utilizado para escribir el código de las Cloud Functions que he implementado.

- **Node JS**: es un entorno en tiempo de ejecución multiplataforma, de código abierto, para la capa del servidor (pero no limitándose a ello) basado en el lenguaje de programación ECMAScript, asíncrono, con I/O de datos en una arquitectura orientada a eventos y basado en el motor V8 de Google. En nuestro caso, nos servirá para desplegar las funciones que creemos con JavaScript y Visual Studio Code en el servidor Firebase Cloud Functions.

- **Postman:** Se trata de una herramienta dirigida a desarrolladores web que permite realizar peticiones HTTP a cualquier API. Postman es muy útil a la hora de programar y hacer pruebas, puesto que nos ofrece la posibilidad de comprobar el correcto funcionamiento de nuestros desarrollos. En nuestro caso, ha sido el programa que he utilizado para lanzar y probar las peticiones HTTP que llamaban a las distintas Cloud Functions.

- **Git:** Es una herramienta para el control de versiones que ayuda a los desarrolladores a controlar diferentes partes y momentos del proyecto para que puedan trabajar en paralelo sin afectar el trabajo de otros. En este caso ha servido únicamente para ir guardando en un proyecto remoto los cambios que se iban desarrollando de la aplicación móvil. En nuestro caso nos ha servido para mantener las nuevas funcionalidades administradas disponiendo de un repositorio donde los diferentes compañeros pudieran acceder al proyecto.

- **Gitkraken:** Es una potente y elegante interfaz gráfica multiplataforma para git desarrollada con Electron. De forma muy sencilla podemos llevar el completo seguimiento de nuestros repositorios, ver ramas, tags, crear nuevos, todo el historial de nuestro trabajo, commits etcétera. En nuestro caso lo he usado, junto con el propio VCS de Android Studio, para actualizar el repositorio de Git.

- **Ninjamok:** Es una aplicación web que nos permite diseñar bocetos de manera rápida y sencilla. Posee una versión de pago donde hay más elementos de diseño y evita marcas de agua, pero para evitar aumentar los costes se ha utilizado la versión gratuita. Esta aplicación fue elegida, frente a otras, debido a que ya conocía su funcionamiento ya que lo he usado para otros proyectos.

- **Diagramas.net (draw.io):** draw.io pro es una aplicación completamente gratis para Google Drive (TM) que te permite dibujar:
    - Diagramas de flujo.
    - LUM (Lenguaje Unificado de Modelado).
    - MER (Modelo Entidad-Relación).
    - Diagramas de Red.
    - Modelos de Procesos de Negocios.
    - Organigramas.
    - Circuitos electrónicos y más.
    - Wireframing y maquetas.

  Características:
    - Cliente nativo HTML 5 con completo soporte para IE 6-8.
    - Enorme biblioteca de plantillas incorporadas.
    - Interfaz intuitiva de arrastrar-y-soltar.
    - Búsqueda de imágenes y función de añadir.
    - Exportación a PNG/GIF/JPG/XML/SVG/PDF.
    - Soporte de dispositivos táctiles.
    - Colaboración en tiempo real.
    - Incorpora diagramas en blogs o wikis.

  En nuestro caso he usado esta aplicación para construir la mayoría de los diagramas, incluyendo los UML, que se pueden ver en este documento que dan soporte a las explicaciones de diferentes apartados.

8. Conclusiones y líneas futuras

Como último punto de esta memoria, voy a dividirlo en dos puntos: las conclusiones que he sacado de todo el proceso que ha llevado el desarrollo y las diferentes evoluciones que puede llegar a tener respecto a funcionalidades y diseño.

8.1. Conclusiones

Este proyecto surgió como continuación de otro que empezó un compañero del máster. Esto implica que no es un proyecto original mío, sino que la especificación y requisitos vienen desde Artefactos y AMFI, y que además se cuenta con una fase del desarrollo ya realizada por un compañero el año pasado, por lo que tuve un periodo de aprendizaje donde tuve que entender qué, por qué y cómo lo había hecho. Por suerte, tuve bastante ayuda de dicho compañero, el cual, no tuvo problema en explicarme y ayudarme en todo lo que necesitase, haciendo esta tarea bastante más sencilla.

A parte del trabajo de aprendizaje con el compañero, también tuve que reunirme con el grupo de "Artefactos" con el que estaba trabajando, ya que la aplicación móvil que tenía que continuar formaba parte de un proyecto donde había involucrada más gente. Con este grupo de trabajadores he tenido varias reuniones, sobre todo al principio del desarrollo, para proponer ideas y ponerlas en común para incluirlas de la mejor manera en el proyecto. Estas ideas siempre estaban orientadas en conseguir que los usuarios de la aplicación tuvieran una buena experiencia usándola y, sobretodo, crear una conciencia social en la que los usuarios de la aplicación como los que no, respetaran las zonas habilitadas para las personas de movilidad reducida y dispersión de aquellas personas que hicieran un uso fraudulento de las mismas plazas gracias al control de ocupación.

Durante estas reuniones se intentaba concretar todo lo posible cómo se debería desarrollar cada funcionalidad que se aprobase para ser implementada. Aunque siempre se buscaba detallar todo lo posible, era bastante frecuente, incluso en reuniones posteriores, que surgieran cambios de diseño que cambiaban bastante la implementación. Esto es frecuente hasta en proyectos como el que estamos tratando, ya que nos basamos en un tema el cual no conocemos, tenemos que investigar aspectos legales, burocracia y protocolos para saber cómo funciona el mundo al cual estamos haciendo la herramienta. Por ejemplo, alguno de los cambios era pasar de tener el correo como dato de acceso al perfil al número de tarjeta de movilidad reducida, ya que nos dimos cuenta que, sobre todo en niños, era posible que quien manejase la aplicación no fuera directamente la persona con problemas de movilidad reducida, sino que pudiera ser algún familiar, por lo que los datos de contacto como el correo podrían cambiar, sin embargo, el número de la tarjeta siempre estaría vinculado con la persona afectada, como ocurre con los DNI.

Cambios así no solo estaban relacionados con el uso y gestión de las tarjetas, sino también teníamos que pensar cómo era la manera más sencilla de que una persona con estas características manejara la aplicación, es decir, que todo los datos y opciones que ofrece la app sean entendibles y fáciles de usar para el usuario.

Cabe destacar que el proyecto trata sobre un sistema que debe presentar una alta tolerancia a fallos, ya que hay muchos factores externos (técnicos y humanos) que pueden proporcionarnos una lectura incorrecta. Por este motivo, se ha hecho un profundo análisis de las diferentes casuísticas que se pueden dar y cómo poder recuperarnos antes posibles fallos.

A medida que iba desarrollando la aplicación tenía que ir planificando la cantidad de funcionalidades y cuánto tiempo iba a tardar en ellas. Esta tarea fue un poco inestable ya que el segundo cuatrimestre del máster fue bastante más intenso, en cantidad de trabajo requerido, de lo que fue el primero, por lo que tuve mucho menos tiempo del esperado para dedicarlo al TFM. A todo esto, se suma el inesperado contratiempo del confinamiento, el cual nos recluyó en casa, haciendo que todo el cuatrimestre fuera más lento e impidiendo las reuniones presenciales con el equipo de desarrollo. Es cierto que, se hizo un gran esfuerzo para poder seguir con el trabajo de forma virtual con videoconferencias, pero personalmente, pienso que siempre se trabaja mejor cuando puedes trabajar en persona con el equipo, sobre todo, teniendo en cuenta que una de las funcionalidades principales del proyecto se basaba en la comunicación con una baliza, por lo que las pruebas que hemos hecho con esto han sido más lentas y tediosas.
A pesar de todo esto, he ido avanzando en el desarrollo de casi todas las ideas que teníamos previstas en el inicio del análisis y las que iban surgiendo por el camino, por lo que estoy bastante satisfecho de haber conseguido un gran contenido para la aplicación.

Por último, me gustaría comentar que he aprendido mucho sobre tecnologías innovadoras para diversas funcionalidades, sobre cómo se comunican entre ellas estoy, y he adquirido un crecimiento profesional y personal para poder adaptarme en medios con los que estoy poco familiarizado. Me refiero, sobre todo, a la adaptación que he conseguido para aprender a entender e integrar herramientas de terceros, como ha sido el caso de las APIs de Google y Firebase.

8.2. Líneas futuras

En este apartado detallaré algunas de las ideas que he pensado para mejorar la aplicación añadiendo nuevas funcionalidades.

- Restringir la funcionalidad de la votación solo a los usuarios registrados y así, asociar el perfil con el voto, para tener un control de quién ha votado qué plaza.
- Construir una aplicación para que un administrador pueda gestionar todo el sistema de forma más sencilla. Actualmente, disponemos de la consola de firebase, pero se queda un poco escasa para realizar algunas tareas de mantenimiento como la verificación de perfiles, entre otras.
- Ampliar la forma en la que el usuario registrado pueda valorar el estado de una plaza, a parte de un “like” o “dislike”, añadiendo un cuadro de texto donde se pueda añadir comentarios o una valoración a través de 5 estrellas, para tener una valoración más exhaustiva.

- Implementar un sistema de verificación del correo para ayudar a la verificación del perfil, ya que así se consigue disminuir los perfiles falsos.
- Implementar un sistema de recuperación de la contraseña por si el usuario la olvida pueda conseguir una y así volver a acceder al sistema.
- Añadir una pantalla donde se muestra un texto indicando la utilización que se hará de los datos proporcionados para el registro del perfil. Además, que se pueda actualizar si la normativa cambia y que el usuario pueda aceptar o no.

9. Webgrafía

1. Firebase - https://firebase.google.com/?hl=es_419
2. Cloud Firestore - https://firebase.google.com/docs/firestore
3. Cloud Functions para Firebase - https://firebase.google.com/docs/functions
4. Cloud Storage - https://firebase.google.com/docs/storage
5. Google Analytics - https://firebase.google.com/docs/analytics
6. Firebase Cloud Messaging - https://firebase.google.com/docs/cloud-messaging
7. Google Maps Platform - https://cloud.google.com/maps-platform?hl=es
8. Especificación del protocolo AltBeacon - https://github.com/AltBeacon/spec
9. Librería Beacon para Android - https://altbeacon.github.io/android-beacon-library/beacon-transmitter.html
10. Activity Recognition Client - https://developers.google.com/android/reference/com/google/android/gms/location/ActivityRecognitionClient
11. IngePark - http://ingepark.es/
12. Equinsa Parking - http://equinsasistemas.com/descargas/SISTEMA%20GUIA.pdf
13. 3PGS Outdoor - https://imasdetres.com/3pgs-outdoor-sistema-de-gestion-de-aparcamiento-en-via-publica/
14. PMUS, Sistema de gestión y control de aparcamiento en vía pública - https://www.esmartcity.es/comunicaciones/sistema-gestion-control-aparcamiento-via-publica-pmus
15. Repositorio de iconos - https://www.flaticon.com/
16. Diagram tool - https://www.draw.io/
17. Noticia sobre las plazas para discapacitados - https://www.diarioinformacion.com/elche/2014/02/19/ayuntamiento-revisara-plazas-aparcamiento/1470936.html
18. Artículo explicativo de los Beacons - https://solidgeargroup.com/beacons-en-android/